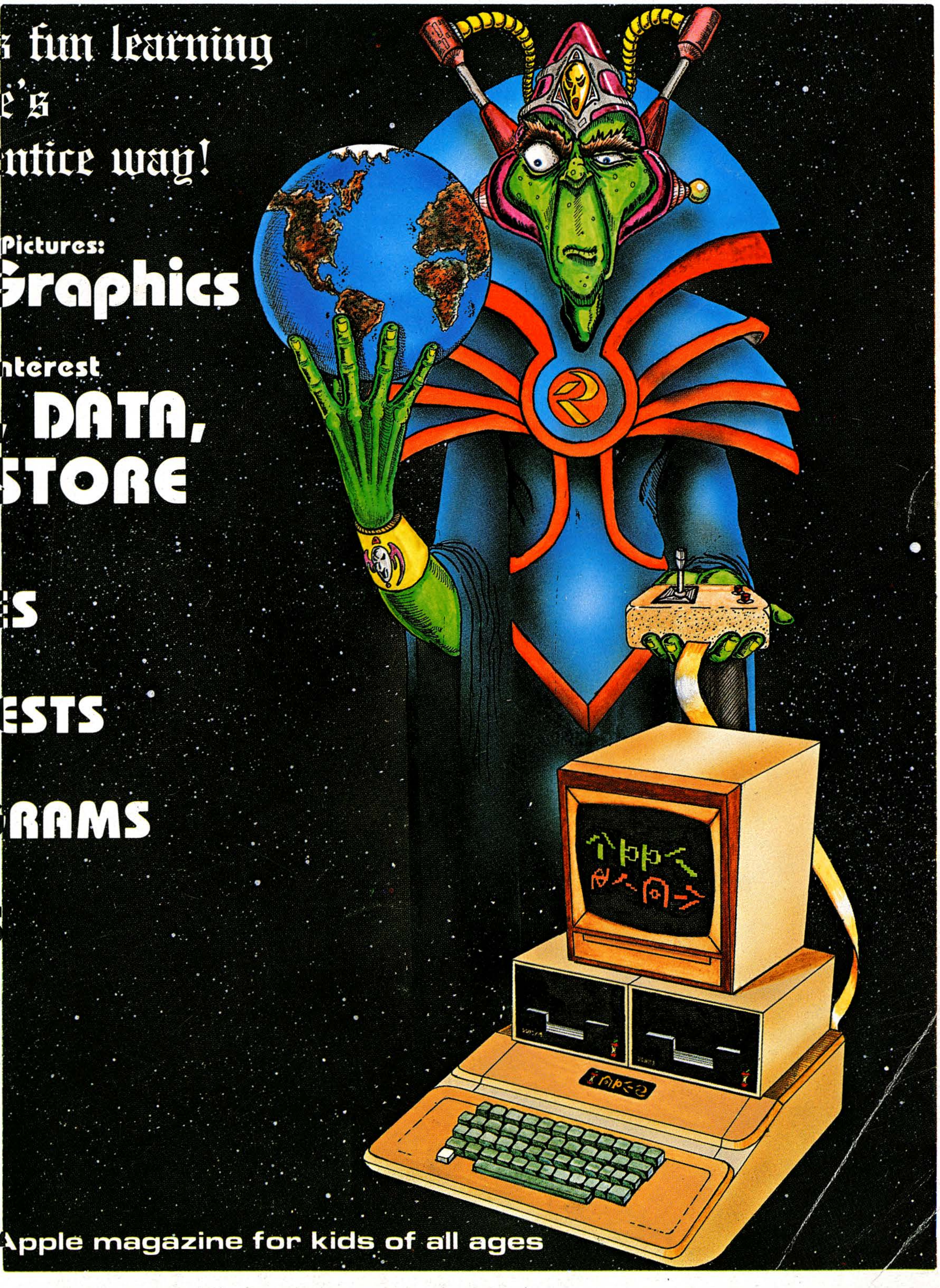
Volume I, No. 3 August, 1984

the APPLE'S APPRENTICE

Learning about Apples $2.50 ($3.25 Canada)



fantasy: it's fun learning . . . the Apple's Apprentice way!

Pretty, Pretty Pictures:

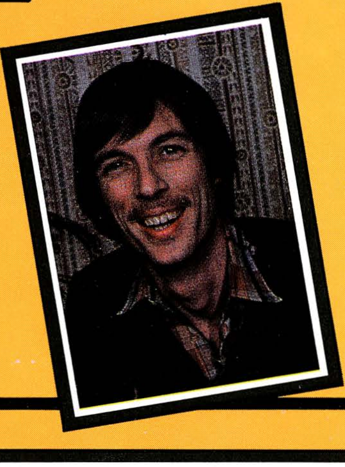
- **MagiGraphics**

Thyng's of Interest

- **READ, DATA, RESTORE**

- **GAMES**
- **CONTESTS**
- **PROGRAMS**
- **HINTS**
- **FUN!**

The Apple magazine for kids of all ages

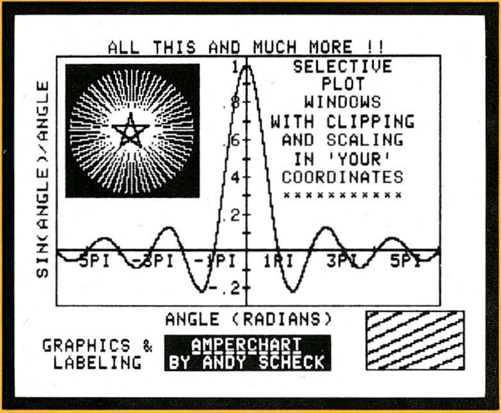


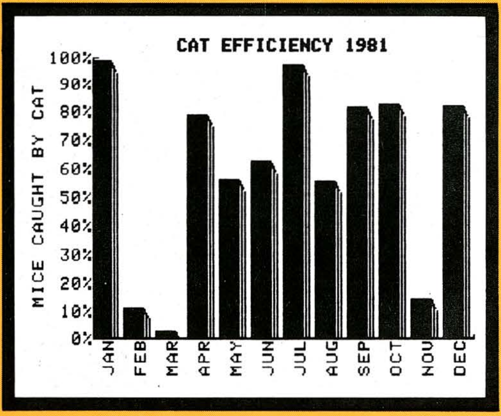
The **only** charting package that supports **Double Hi-Res** from

Roger Wagner Publishing, Inc.

"If you want to have the freedom to create **your own** custom charts and other Hi-Res graphics, then the Routine Machine &Chart is your key to making your Apple do what **you** want."

—*Roger Wagner, President*


ALL THIS AND MUCH MORE !!
SELECTIVE PLOT WINDOWS WITH CLIPPING AND SCALING IN 'YOUR' COORDINATES
****
1
.8
.6
.4
.2
-.2
-5PI
-3PI
-1PI
1PI
3PI
5PI
SIN(ANGLE)/ANGLE
ANGLE (RADIANS)
GRAPHICS & LABELING
AMPERCHART BY ANDY SCHECK



FREE BONUS!
Easy-Chart demo program creates line, bar & pie charts from your own data!

ROUTINE MACHINE &CHART™

"Not just for programmers anymore..."

Once upon a time, only professional programmers could take full advantage of what a computer could do. Now, we've made things easy for you too! With the RM &Chart you'll be able to create charts and other Hi-Res graphics on your own Apple, the way **you** want them!

The RM &Chart package includes a two-sided diskette packed with fantastic ways for you to use **both normal and double high resolution graphics.** With RM &Chart, anyone can add **custom graphics commands** to ordinary Applesoft to create one of the most powerful graphics systems imaginable for the Apple computer. And it's easy too! Adding a RM &Chart command to your own programs takes **literally less than a minute!**

Routine Machine &CHART gives you:

- Complete charting commands including automatic scaling and axes generation; even Log scaling!
- Hi-Res labels (in regular or double Hi-Res graphics) to add text to any picture.
- Regular & Double graphics printing for Epson printers.
- 3D graphics commands for any rotation & scaling.
- Special "window" routines to do Hi-Res windows; even pull-down menus like the Macintosh!

PRICE: Only $39.95!

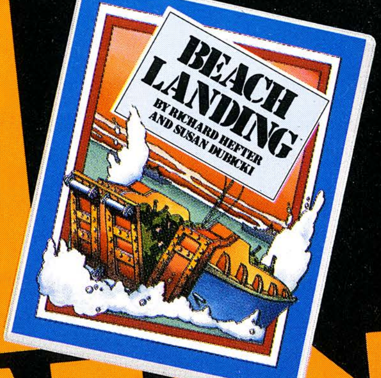
Compatible with Apple II/II+, IIe and IIc computers.
Plus, Routine Machine &Chart is copyable & listable!
(Double Hi-Res features require an Apple IIc or Apple IIe with EXTENDED 80 column card)

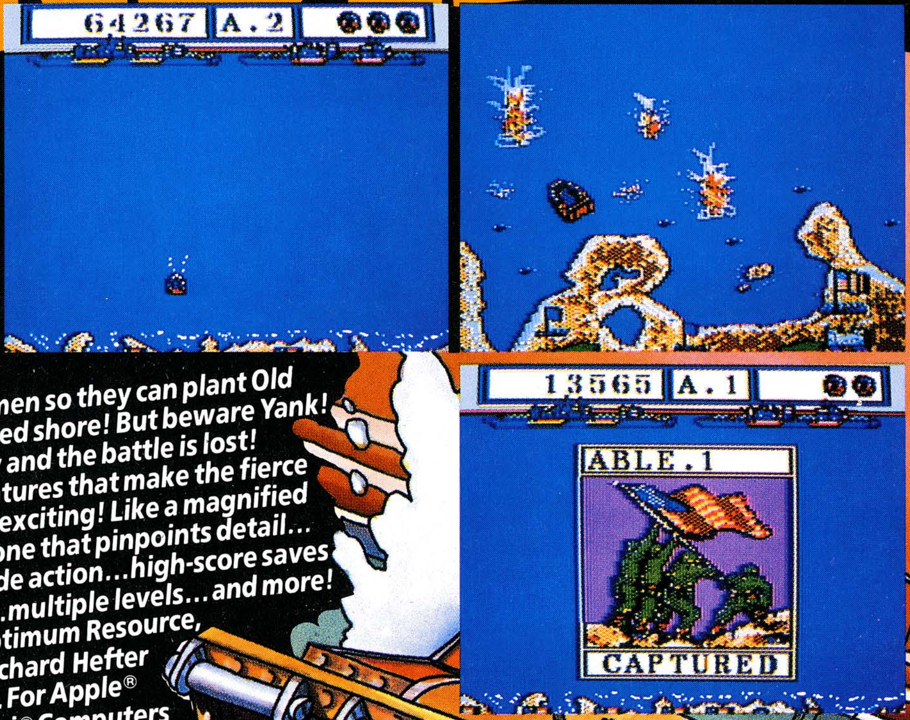
Ask for these easy to use RM packages: RM &Array, RM &Screen, RM &Sampler I, and RM &Sampler II (formerly Routine Machine)

Roger Wagner
PUBLISHING, INC.

10761 Woodside Avenue • Suite E • P.O. Box 582 • Santee, CA 92071 • 619/562-3670

BEACH LANDING!



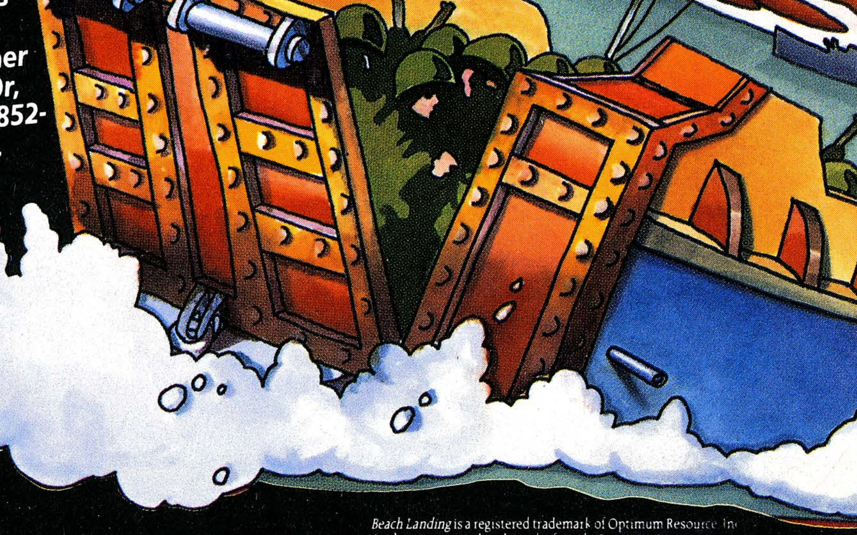


Beach Landing, the spine-tingling new strategy arcade game from Weekly Reader Family Software, offers the thrill of combat as landing ships hit the shell-wracked beach …dodge withering enemy fire…and unload our fighting men so they can plant Old Glory on the embattled shore! But beware Yank! One error in strategy and the battle is lost!

Look for great features that make the fierce invasion even more exciting! Like a magnified view of the battle zone that pinpoints detail… real strategy…arcade action…high-score saves …fabulous sound…multiple levels…and more!

Developed by Optimum Resource, Inc. Designed by Richard Hefter and Susan Dubicki. For Apple® Computer and Atari® Computers with 48K and one disk drive.

Look for Beach Landing in finer computer stores everywhere. Or, order by calling toll-free 1-800-852-5000, Dept. AE-62. Only $39.95.



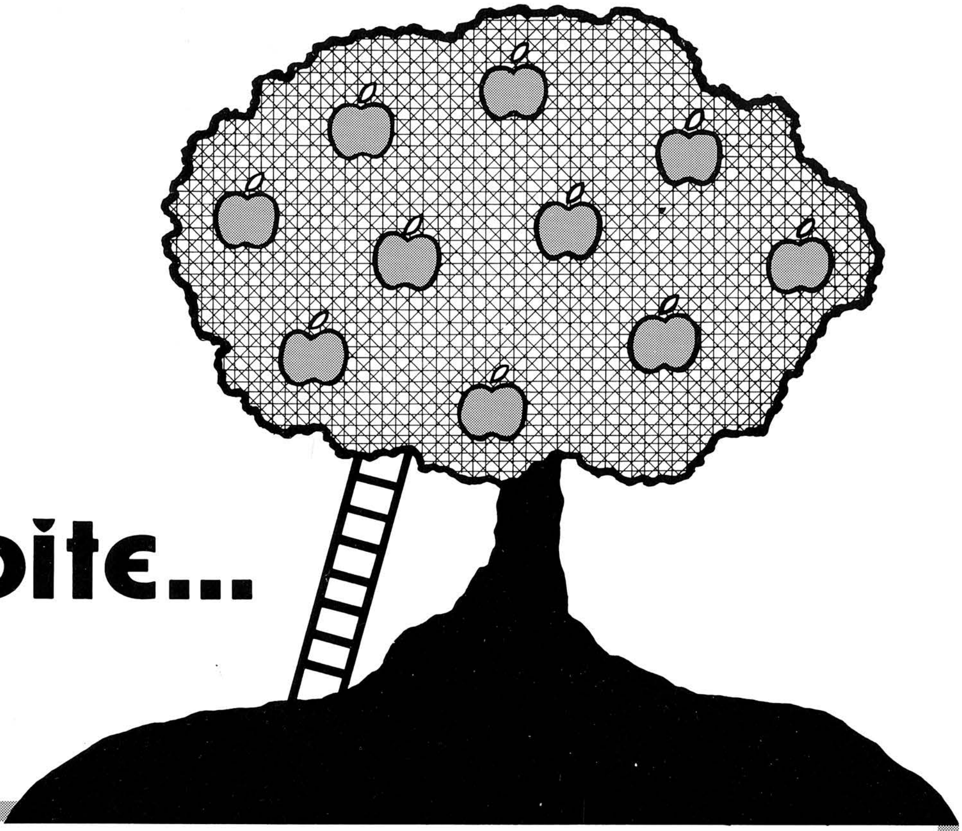
BEACH LANDING

Weekly Reader Family Software

A division of Xerox Education Publications
Middletown, CT 06457

Beach Landing is a registered trademark of Optimum Resource, Inc. Apple is a registered trademark of Apple Computer Inc. Atari is a registered trademark of Atari Computer Inc. A M44AE62

Take a bite...



...of Apple Orchard!

Your Apple will love you for subscribing to APPLE ORCHARD Magazine, the premier magazine for Apple owners!

APPLE ORCHARD has all the information you need for your Apple. Our hardware and software reviews are done by professional computer people. Our NEW PRODUCT section is the largest in a monthly publication. In addition, there are articles that let you know what people are doing in business, finance, education and at home. We make sure that our utilities, diagnostics, and programming tools are easy to use.

You and your premier computer deserve the best. Subscribe today and save 36% over the newsstand price. Why not give a gift to a deserving Apple owner?

Please enter a subscription to APPLE ORCHARD for:

NAME ______________________________

ADDRESS ______________________________

CITY ____________ STATE ________ ZIP ________

Annual Subscription Rate: U.S., FPO, APO–$24.00/1 yr., $45.00/2 yr.
Canada: $30.00/1 yr., $55.00/2 yr.; Foreign: $36.00/1 yr.
Write for Airmail rates.

☐ Check or money order enclosed (U.S. Funds) ☐ VISA ☐ MasterCard

Card No. ______________________ Exp. __________

Signature ______________________________

Reply to:

APPLE ORCHARD SUBSCRIPTIONS
P.O. Box 6502
Cupertino, CA 95015

Subscribe today! You. . . and your Apple. . . won't want to miss a single issue!

Volume I, No. 3 August, 1984

the APPLE'S APPRENTICE

The Apple's Apprentice is published by:

Emerald City Publishing, Inc. P.O. Box 582-AA, Santee, CA 92071, (619) 562-7785. Entire Contents copyright © 1984 by Emerald City Publishing, Inc.

Roger Wagner, publisher
Val J. Golding, editor

Donna Sexton
Advertising Director
Pamela Lambert
Circulation Manager
Linda Anderson
Illustrator
Margot and Al Tommervik
Godparents

Subscriptions: $24/12 issues, $46/24 issues

Advertiser's Index

ArtSci	BC
Beagle Bros.	16,17
Emerald City Publishing	5, 27
Hi Tech of Santa Cruz	22
Original Computer Camp	32
Apple Orchard	2
Weekly Reader Family Software	1
Penguin Software	IBC
Roger Wagner Publishing	IFC
Shenandoah Software	30

The Apple Magazine
For Kids of all ages

Cover art by:
Mike Wilcot and
Doug Westercamp

Scene from *Bezare*™
A trademark of Roger Wagner Publishing, inc. Reproduced through their courtesy.

Table of Contents

From the Tower 4
the Apprentice Wizard
a view of these pages through the wizard's magic spyglass

READ, DATA and RESTORE:
Thyng's of Interest 5
Michael Thyng
right from the hip: how to use these statements

Pretty, Pretty Pictures:
MagiGraphics 9
David Sparks
converting cousin ricky's spirograph to the apple

The Duncan Letters 12
Joe Holt/Laurie Wofford
the sorceror confesses he had the castle wired for electricity

Assembly Language for Teens:
The Sourceror's Apprentice 14
Mike Newton
using your first monitor commands

Ask the Wizard 18
The Wizard of Fairhill Castle
fixing renumber, game magazines and cataloging deleted files

IF/THEN/ELSE:
Spells and Potions 19
Ralph H. Swerdlow, M.D.
simulating the else command in applesoft

Uncle Bill's Column 21
Uncle Bill
better get some adhesive tape to hold your sides in

Meet the Turtle:
Logo Lingo 23
Jeff Sandys
moving the screen turtle and setting hi-res colors

The Wizard and the Princess:
Shortcut to Adventure 25
Cassidy/Katz/Lynn/Waisman
a 40 room guided tour, plus a map

Spanish Verb Conjugator 27
Tim Graham
the apple can do your spanish homework

Game News and Reviews
Ye Olde Game Shoppe 28
Val J. Golding, prop.
bezare and hinky-pinky are featured

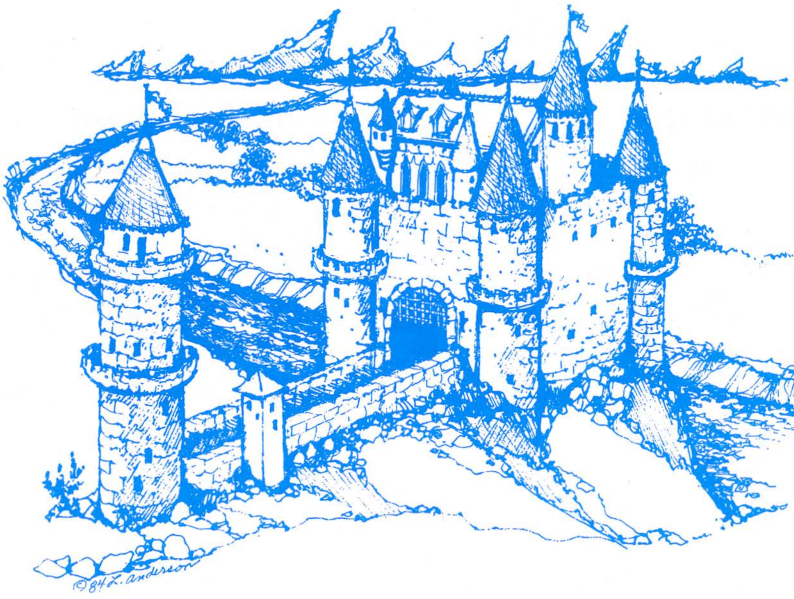
Cross Words 29
last time was so much fun, let's do it again

New Products:
The Crystal Ball 31
learning aids for dyslexia are included

Contests Galore:
Daze of the Knights 32
David Sparks
two terribly tempting teasers for talented thumb twiddlers

Dija Know 32
the deja vu of computer hints . . . oh well, just look . . .

From the Tower



the Apprentice Wizard

Welcome. We're glad to see you have joined us again. We're always happy to see our old readers back, along with the new ones. For the benefit of the new readers, a word or three on what **Apple's Apprentice** is all about.

The Apprentice is a different breed of Apple magazine. It is for kids, but in addition, for kids of all ages. Or, as we like to say, "for kids from 8 to 80." Mainly it's a place where you can drop your hat, put your feet up on a chair, relax and **learn** about Apple Computers in an informative, fun way.

A lot of magazines **say** they cater to beginners; we **do**. You don't have to know much at all about Apples to read and enjoy the Apprentice and learn about them from stories written in a loose, breezy style that don't push you into the high-tech world on page two. What's more, as we progress, we will regress. That is, somewhere along the way we'll pause and review or rewrite some of our earlier beginner's material.

Best of all, you don't have to learn about Apples at all if you really don't want; you can enjoy the game reviews, the comics, puzzles and contests or look in the **Crystal Ball** to see what the newest in peripherals and software is. On the other hand, you might like to wander through [and maybe even get involved in] stories like Joe Holt's fantasy **The Duncan Letters**. You might even find out that programming can be fun. Graphics is accounted for as well, with **Magigraphics:** Pretty, Pretty Pictures. Talented author David Sparks tells you all you ever wanted to know about Spirographs, and cycloids, as well.

For those who want to learn programming, we offer several delicious flavors, all easy to take. This month we introduce **Uncle Bill's Column**, filled with gems of humor and handy hints. **Thyng's of Interest** also premieres this time, with author Michael Thyng telling the truth, the whole truth and nothing but the truth on how to use DATA, READ, and RESTORE commands. Ralph Swerdlow has come up with a way to simulate the IF/THEN/ELSE statement in **Spells and Potions**, and if you really want to get into it, **The Sourceror's Apprentice** will take you teenagers and adults by the hand and gently guide you into the realm of assembly language programming.

By the way, speaking of the Sourceror's Apprentice, author Mike Newton included a little chart of ASCII screen display characters to go with his story. It's mighty small, you'll admit, and if we could have made our printer print even smaller, we would have. Why? So we would have an opportunity to tell you about our ASCII wall chart. In addition to the data in the small chart, we've added a brief explanation of why ASCII is different in Applesoft than in the monitor. How can you get it? That's the easy part. It's all yours, postpaid, for just $1.50 [deduct 25% for multiple copies]. Send to:

ASCII Chart
P.O. Box 582-AA
Santee, CA 92071

A funny thing happened to us on the way to San Diego . . . Many of you may know us as the former editor of **Call -A.P.P.L.E.** and in fact may have read the story on "Val's Basement," along with its photographs, in the June issue. The long and short of it is that literally a ton of documentation, disks, programs and books was packed into our decrepit VW beetle, probably a rare example of programs in a bug, rather than the other way to.

Oh well . . .

Last on the agenda [well, almost]. Reader participation. **The Apple's Apprentice** can only be **your** magazine with your help and participation. We have three features specifically for that purpose: **Back Talk** [letters, round table discussions, etc.], **Ask the Wizard** [The Wizard of Fairhill Castle answers all programming questions, none too easy, none too hard], and **Daze of the Knights** [witty contests to tax your wits and programming skills]. So let's hear from you guys and gals out there, and that includes comments on what you'd like to see in **The Apple's Apprentice**, as well.

Almost as a postscript, don't forget you can receive 12 fun-packed issues of **The Apple's Apprentice** for just $24.00; that's only $2.00 per issue. Note that we said 12 **issues**. That's because we started in April, then skipped a month, but every subscriber can be assured of receiving their full subscription and their money's worth. Now it is past the witching hour, so until next time, we'll leave you with this thought: **On a clear disk, you can seek forever.**

READ, DATA and RESTORE:

Thyng's of Interest

Michael Thyng

One feature of Applesoft that is commonly under–utilized is the DATA statement. The DATA statement is one of the common methods of entering working data into a computer program. Its advantage is that you can count on the same data time after time. Its disadvantage is that it is only sequentially accessible. Lets look at its structure, then try out some examples.

You prepare the data you're going to use by keying in a line number, the command word DATA, followed by the actual data you want to use. The data does not have to be in any particular order to meet the Applesoft requirements. So you enter it in the order for your specific programming needs. For numeric data, a typical line of code looks like:

```
200 DATA 34,16,73,106,4
```

I chose line 200 for my DATA statement. I could have chosen line 5, or line 62000. I can put the data in the beginning, middle or end of the program. I can use any of the valid line numbers that any other Applesoft command can use. It doesn't matter. Your program will find the data. The only thing you really should do every time you use DATA statements, is to bunch them together so they are easier to find. Anything that's easy to find is faster to modify if and when that becomes necessary.

DATA statements may be placed anywhere in the program

You can have more than one DATA statement. In fact, you can have a program of almost all DATA statements. The only limit I have ever found is that you are limited by the amount of memory in your computer. For most applications this means you will never run out of DATA space. Let's look at variations of line 200 that are its equivalent. Instead of writing line 200 as it is, we could have substituted:

```
300 DATA 34,16
310 DATA 73,106
320 DATA 4
```

Or, we could have put each piece of data in a separate DATA statement as in:

```
410 DATA 34
420 DATA 16
430 DATA 73
440 DATA 106
450 DATA 4
```

As you can see, you can mix and match the data you are keeping in DATA statements in any logical way you see fit. Also, note that I just used 5 data elements in my line 200 DATA statement. You could use 10 or 20 or more. In fact, the only limit imposed on you is the 238 character entry limit of an Applesoft programming line.

Okay, now we've got the data stored. But you can't work with it just yet. You have to have a way to access it. You access the data by using a READ command. The READ command looks like this:

```
700 READ R
```

When the computer is running a program which encounters our line 700, it will immediately do a search for a DATA statement. Actually, it will look for the first DATA statement in your program. The first DATA statement in your program is, of course, the one with the lowest line number. DATA statements may be placed anywhere in the program, but tradition usually places them at the end. In some ways, DATA statements are like REM statements in that the Applesoft interpreter will completely ignore them, except when it is executing a READ.

I've been consistent to use the same data and in the same order in our ex-

Savings for Apple's Apprentice Subscribers!

Want to solve the unsolvable? Read *Ye Olde Game Shoppe* on page 28 and learn the mystery of the *Bezare* game. You can save $5.00 from the retail price *and* get a free poster for just $14.95. Or, you can get an information-filled ASCII wall poster for only a buck and a half if you're an *Apple's Apprentice subscriber. It's as easy as filling in and mailing the coupon below.*

Apple's Apprentice
P.O. Box 582-AA
Santee, CA 92071
(619) 562-7785

Please send the following items:

☐	ASCII Wall chart	$ 1.50	______
☐	4-color Bezare poster	$ 5.95	______
☐	Bezare game (only)	$12.95	______
☐	Bezare game and poster	$14.95	______
☐	12 issue subscription	$24.00	______
☐	24 issue subscription	$46.00	______
	Total	$______	

☐ Visa ☐ Mastercard ☐ check

Card No. ______________________

Exp. date ______________________

Name ______________________

Address ______________________

City ______________ State ______ Zip ______

Signature ______________________

(if minor, parents signature)

amples, so we shouldn't have much confusion. The first piece of data that is read into R is 34. From this point on, numeric variable R has 34 as its value. Why did I choose R as the numeric variable? Only because it will be more meaningful in the examples I will use in this article. I was free to use any numeric variable that follows the syntactic rules of Applesoft.

Now we're going to use the value of R in a formula to calculate something. For this case, let's figure out the area of a circle. The formula is PI times the radius of the circle squared. In Applesoft, it would look like:

```
800  PI = 3.1415926
900  A = PI * R * R
1000 PRINT "RADIUS ";R;" AREA
     ";A
```

What you will see output is: **RADIUS 34 AREA 3631.68105** — That used our first numeric value in the DATA statement. Suppose we run the program again. What value will be used this time? Did you say 16? Wrong! The answer is 34. Run it and see. The reason is that each time you run the program, a certain pointer inside the computer points to the first value following the first DATA statement in any program. How do you get to the data elements past the first one? You need to execute the READ command again.

By the way, I have been interchanging the words command and statement. Generally, they are interchangeable, but statement seems passive, as DATA statements are, and command seems active, as READ commands are. Now, back.

To get the program to READ again, we just have to GOTO line 700 again. We'll add line 1100 to our program.

```
1100 GOTO 700
```

At this point you should have a program with one set of data in it. Depending on which option you chose to hold your data, you should have something like I have below. I chose yet another variation in the ways to keep data in DATA statements to further emphasize the flexibility of options you could use.

```
200  DATA 34
220  DATA 16,3,106
240  DATA 4
700  READ R
800  PI = 3.1415926
900  A = PI * R * R
1000 PRINT "RADIUS ";R;" AREA
     ";A
1100 GOTO 700
```

RUN the program. Now you get all five numeric values out as well as the areas calculated for all five. Congratulations. But the kudos are short lived. You also get an **?OUT OF DATA ERROR IN 700** message. This means your program tried to read more data than there was available. Let's see how reading again and again accesses different data. Each time you READ data, you get the data element that the pointer I talked about earlier is pointing at. When the computer executes the READ statement the first time, the pointer points at 34. It points at 34 because it is the first data element in the first DATA statement. As soon as it does the READ, the pointer moves to the next data element. That way, it is always ready to READ the most currently available piece of data. So after the computer has done three READs, the pointer is pointing at the fourth piece of data. In our example, it points at 106.

What happens if you try to READ more data than is in the program?

What happens if you do more READing than you have data? That's what we did. We tried to read six times, but we only had five data elements. After five READs, the pointer had adjusted itself to point at the sixth piece of data. But there is no sixth piece of data. That in itself is okay. It's only when we try to READ a sixth time that we encounter a problem. Obviously, if we try to read data and there is no data, something has to be faulty. I mean computers are supposed to be logical devices. (Aren't they??) Some people might attempt a band-aid type solution to this situation by using an ONERR GOTO. This is a poor programming practice. (You will see this done even in commercially available programs, but what can I say.) No, the correct solution to the problem is to find the last piece of data and either bring the program to a logical conclusion or go on to another place in the program. We can accomplish this in at least two ways.

The first way is to know what the exact value is of the last data element in your list of data. Then test for that value and, in our example, end the program. Our last data element is 9. We would test for that with an IF statement. The logical place would be just before the GOTO. We want to have read all the data, have done all the calculations, and have displayed all the output. Then when we test, we are going to prevent the computer from going back to do another READ.

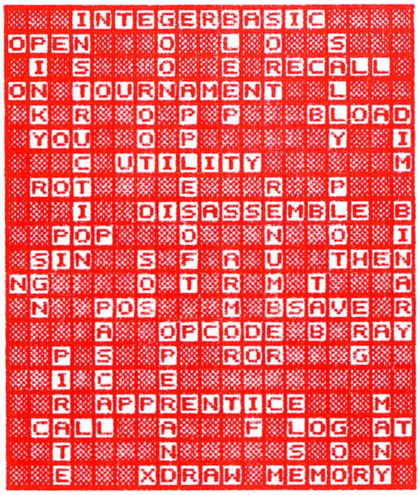
```
1020 IF R = 4 THEN END
```

This way presents a problem if our last data element ever changes. You see, if the fifth data element is 8, and we are testing for 4, then the computer won't end, and we'll go back to read a sixth data element, run out of data, and the program will break in 700 again.

. . . add a special last data element

The second way, and a way I always use, and highly recommend, is to add a special last data element. This last data element must be a value that is never going to be found in the working data. I usually use 99999. In all applications with which I've worked, I've never come across 99999 as a legitimate piece of working data. If, in your appli-

Solution to puzzle on page 29



```
###INTEGERBASIC####
OPEN###O##L#O##S###
#I#S###O##E#RECALL#
ON#TOURNAMENT##L###
#K#R##O#P#P###BLOAD
#YOU##O#P######Y##I
###C#UTILITY######M
#ROT####E###R##P###
###I##DISASSEMBLE#B
##POP###O###N##O##I
#SIN##S#F#A#U##THEN
NG####O#T#R#M#T###A
#N##POS###M#BSAVE#R
####A##OPCODE#B#RAY
##P#S##P##ROR###G##
##I#C##E###########
##R#APPRENTICE###M#
#CALL##A###F#LOG#AT
##T####N#####S#O#N#
##E###XDRAW#MEMORY#
```

cation, you found that 99999 might be valid data, you would just choose a final value that wouldn't be. To use this method, pull out line 1020 and enter:

```
720 IF R = 99999 THEN END
```

Why did we have to move the IF? Line 1020 tested the data *after* we had used it. What we need with the code data is to test *before* we use it. With line 720, our program reads the data and before any piece of data can be used, it is checked for the value 99999.

One of the uses for the concept of the DATA statement is in something called edit checking. Suppose you wrote a program to enter the data to pay bills by check. You would probably have a need to enter the payee's name, today's date, the amount of the check, and some expense account number. You probably have certain expense account numbers that are valid for your company or home. You only want to input valid expense account numbers. But people make mistakes, so you know that from time to time, someone is going to key in data that "just ain't right." Now comes the DATA statement's use. All the values in the DATA statement are exactly your expense account numbers — except the last one, which is 99999. After each input for an expense account, you read the valid expense account numbers from DATA statements, and compare them against the expense account number that was input. If the first one doesn't match, you read again and keep reading until you find a match or until you reach 99999. If you reach 99999, and you know you have every valid account number stored in DATA statements, then it means the expense number entered through the INPUT statement was wrong, and should be re-entered.

A portion of the code to enter and check expense account numbers would look like this:

```
100  INPUT "ENTER EXPENSE
     NUMBER (9999) ";EX
300  READ X
500  IF X = 99999 THEN PRINT
     "NO FIND":GOTO 100
600  IF X = EX THEN 1000
700  GOTO 300
800  DATA 1018,1020,1030,1050
810  DATA 3000,3010,3030,3070
820  DATA 4002,4020,4050,4060
890  DATA 99999
1000 REM CONTINUE ON WITH
     THE PROGRAM
2000 PRINT "FOUND ";X
2020 FOR DE = 1 TO 1200:NEXT
     DE
2100 GOTO 100
```

To check this out, why don't you run it and enter valid account number 3010. You should see it displayed with the message: **FOUND 3010**. Now try 4050. It works too. Just one more. Try 1020. It's on our list. Oops, what happened? We got a message **NO FIND**. That suggests that we got to the end of our data. Well, we did. Remember the internal pointer? Where was it after we found 4050? It was pointing at 4060. When we input 1020, the program read 4060. The pointer advanced to 99999. When the program discovered that 4060 and 1020 didn't match, it went back to read another value. The only value left was 99999.

Wouldn't it be nice if we could manipulate the pointer so we could look at the same set of data more than once? Guess what — we can. There is a command called RESTORE. This causes that pointer (no, you can't ignore it) to place itself in the same position it had just after you typed RUN. That means we can read the data from the beginning anytime just by using

. . . we can READ data from the beginning by using RESTORE . . .

```
]LIST

10  REM

MULTI-USE DATA STATEMENT DEMO

       MICHAEL THYNG

100  DATA DAYS
110  DATA SUNDAY,MONDAY
120  DATA TUESDAY,WEDNESDAY
130  DATA THURSDAY,FRIDAY
140  DATA SATURDAY
150  DATA END
200  DATA PASSWORDS
210  DATA LEO,JOHN,LOUISE
220  DATA EVERETT,ARLEAN,JOAN
230  DATA END
300  DATA MEDICAL
310  DATA EB25-X,T3-00SL.34,DD25
320  DATA W99,W89,W96,W12,W14
330  DATA M10,M20,M30,M40,M60
350  DATA END
400  DATA EXPENSES
410  DATA 1018,PAYROLL
420  DATA 1020,OFFICE EXPENSE
430  DATA 2000,RENT
440  DATA 2500,VEHICLE FUEL
450  DATA 3000,WELDING
460  DATA 5000,COFFEE
470  DATA 7800,LEGAL EXPENSE
490  DATA END
1000  REM
PASSWORD INPUT AREA

1100  INPUT "ENTER PASSWORD ";P$
1200  RESTORE
1300  READ PC$: REM
LOOK FOR "PASSWORD"
1400  IF PC$ < > "PASSWORDS" THEN
     1300
1500  READ PC$
1600  IF PC$ = "END" THEN  PRINT
     "INVALID USER ACCESSING SYST
     EM.": PRINT "SUMMON HELP": END

1700  IF P$ < > PC$ THEN 1500
1900  REM
NOW SYSTEM HAS VALID USER

2000  REM
DAY INPUT AREA

2100  INPUT "ENTER DAY OF WEEK ";
     W$
2200  RESTORE
```

RESTORE. Let's put it in the program. First let's try it in line 400.

```
400 RESTORE
```

Running the program again, we'll start our test trying to find 3010. Something different happens. It looks like the program went away and forgot us. It didn't stop doing things. As a matter of fact, it is doing a lot of things — the same things over and over. It is reading the first data element 1018. That automatically causes the pointer to point at 1020. Then in line 400 we restore the pointer to 1018. There is no match of 99999 or 1018, so program control goes back to line 200 to read another value. The pointer is currently pointing at 1018. Every time we read, we get 1018. We can't have the RESTORE in line 400. We could have it in line 200 instead. This is the right location for RESTORE. This is the most basic use of the DATA statement.

We just read one variable at a time with our READ statement. If we were going to use DATA to store X and Y coordinates for some application, we could read them with a line like this:

```
2300 READ X, Y
```

It will require that your data be considered as pairs of values. Each time you do a read, you are accessing the next two numbers. Don't forget that the last data, 99999, also needs a second 99999. If you don't do that, X would get a value and Y would cause an out of data error.

The data statement can also be used with alphanumeric data. The only major change, if you can call it major, is to change the variable to a string variable. If we went back to the expense account number example, with modifications, we could access the number and the description of the number. The data would look like this:

```
1000 DATA 1018,PAYROLL
1100 DATA 1020,OFFICE
     EXPENSES
1200 DATA 1030,RENT
1900 DATA 99999,99999
```

To read this data, the READ statement would need to be modified to look like:

```
12000 READ ACCT,DESC$
```

I'd like to show you a trick that I learned. I had a need to use the DATA statement for more than one type of data. That is, we used it for expense account numbers here. Well I had a need to check about four different things and had to find a way to read each set of data differently. I wanted to check the days of the week against a DATA table (that's what we've been making). I wanted to check expense account numbers. I had to check out who was operating the system against a list of those who were acceptable, and I had to check a list of special medical codes that were susceptible to entry errors. The problem was (is) that the days of the week, the passwords, and the medical codes just require one alphanumeric entry and the expense accounts require two entries — a numeric and an alphanumeric. Besides that, there is also the problem of telling the computer which group of data is which.

Where do the passwords end and the medical codes begin? What I did is label each set of data with yet another entry. At the beginning of the days of the week, I put in an entry of "DAYS." Just before the passwords, I put in "PASSWORDS". Before the medical data I used "MEDICAL," and before the expenses, "EXPENSES". To use this setup, I had to have a few lines of code that just check for the four code words. But after I have detected the code word for which I'm looking, I can branch right to the proper read routine to access the data. I think this program will show you how it works.

```
2300  READ WC$: REM
LOOK FOR "DAYS"
2400  IF WC$ <  > "DAYS" THEN 230
      0
2500  READ WC$
2600  IF WC$ = "END" THEN  PRINT
      : PRINT "WRONG DAY": PRINT :
      GOTO 2100
2700  IF W$ <  > WC$ THEN 2500
2800  REM
DAY OF THE WEEK

2900  REM
NOW SYSTEM HAS VALID
DAY OF THE WEEK
3000  REM
MEDICAL CODE INPUT AREA

3100  INPUT "ENTER MEDICAL CODE "
      ;M$
3200  RESTORE
3300  READ MC$: REM
LOOK FOR "MEDICAL"
3400  IF MC$ <  > "MEDICAL" THEN
      3300
3500  READ MC$
3600  IF MC$ = "END" THEN  PRINT
      : PRINT "WRONG CODE": PRINT
      : GOTO 3100
3700  IF M$ <  > MC$ THEN 3500
3900  REM
NOW SYSTEM HAS VALID MEDICAL CODE

4000  REM
EXPENSE ACCT INPUT AREA

4100  INPUT "ENTER EXPENSE ACCT "
      ;A
4200  RESTORE
4300  READ AC$: REM
LOOK FOR  "EXPENSES"
4320  REM
NOTICE ONLY THE ALPHANUMERIC
READ FOR AC$
4400  IF AC$ <  > "EXPENSES" THEN
      4300
4500  READ AC,AC$
4600  IF AC$ = "END" THEN  PRINT
      : PRINT "WRONG CODE": PRINT
      : GOTO 4100
4700  IF A <  > AC THEN 4500
4900  REM
NOW SYSTEM HAS VALID EXPENSE DATA

5000  REM
REST OF THE PROGRAM GOES HERE.
```

Pretty, Pretty Pictures:
MagiGraphics

David Sparks

Cousin Ricky appeared to have it better than I did. At least I thought so while we were growing up. His sandbox had a roof. No matter what house he moved into, his backyards were wonderfully large, wooded, and sometimes steeply sloped. I was always glad to go for a visit. Ricky's room contained lots of the most amazing toys. And of all the toys, one in particular drew me back and back again: his Spirograph.

Do you know what a Spirograph is? A box full of plastic circles with gear teeth inside and out and pencil holes in the rim. You thumbtack one circle to a piece of paper. Engage its teeth with those of a second, unattached circle. Put a pencil point through one of the holes in the movable circle. Go around and around as long as you like, making the most amazing patterns.

You could put a tiny circle inside a bigger one and make patterns in the middle. You could make patterns outside a circle. You could do both. You could draw one pattern, then another over the top of it. You could have different colors. You used up all the paper the grown-ups allowed long before you ran out of ideas. In an age without video games, these splendid figures set the light dancing in my childhood eyes.

Grown-ups, it is well known, cannot tolerate childlike enthusiasm very long. It makes them quite uncomfortable. It's bad enough when the enthusiast actually is a child, but let someone Big get caught up in the sheer joy of a thing? Oh dear! It just isn't done.

To play with Spirograph pictures, grown-ups have to call them something else. Something terribly dreary and Important-Sounding. Either that or something trendy and high-tech, perhaps hinting of robots or the Space Program. And definitely, something which can be done using a Computer. Something which sounds Terribly Difficult, if you please, but actually is quite simple.

We're only too happy to oblige. Say hello to *Epicycloids* and *Hypocycloids.* They are, respectively, the Spirograph pictures you draw outside the fixed circle, and inside. This concept is illustrated in the accompanying box.

Good tools already exist to turn your Apple into an electronic artist of astonishing capacity, so I'll just tell you what they are, and give you an example program showing how to use them.

In the February 1984 *Byte*, an article by Robert and Ted Sussman described a method for calculating computer plotting points of epicycloids and hypocycloids. Their work was adapted from Murray Spiegel's *Mathematical Handbook of Formulas and Tables*, McGraw-Hill Schaum's Outline Series, 1968. Their formulae, along with some definitions of various terms in this article, appear at the end of this article.

For the program listed here, I have converted the Sussman formulae into Applesoft Defined Functions, one of the little-known, little-used bits of "magic" your Apple can perform. With these particular functions, the program computes the way to draw spirographic pictures on your screen.

While you can use the program without needing to know about Apple Hi-Res graphics, let's discuss how the Hi-Res screen works. Apple Graphics consist of thousands of tiny dots. Hi-Res graphics give you control over each dot individually. You can make it be any one of four colors, plus white. Black is a color, too, which we use to "turn off" the dots. You set the color with the Applesoft command **HCOLOR=X** where X is a value from 0 through 7. See your Applesoft manual.

To "turn on" a dot you select the color then use the Applesoft command, **HPLOT X,Y** where X and Y are the *screen coordinates* of the dot. X values can range from 0 to 279. The lower the number, the nearer to the left edge of the screen. Y values can be from 0 to 191. The lower the number the nearer to the *top* of the screen. **HPLOT 0,0** would activate the dot in the upper left corner. **HPLOT 279,191** would affect the dot in the lower right.

Now, most of us have drawn graphs in math class at school, and the arrangement of the Hi-Res screen can be a bit vexing at first. We're used to the 0,0 location, also called the *origin*, being the lower left corner, not the upper left. Why is the Apple different? For some very complex reasons, it was easier for the designers of the Apple to do it that way. They figured, hey, it's like a bicycle — looks complicated at first, but once you get the knack of using it, then it's easy and a lot of fun.

Let's talk about *aspect ratios*. When I first wrote this program, my pictures came out as ovals. For a while this was alright. Cousin Ricky's Spirograph never gave *him* ovals! But I felt there should be a way to control things, even to get that special oval known as a circle if I wanted it. There were two approaches, one to add complexity to the Defined Functions, the other to simply adjust the scaling on the screen. I chose the latter.

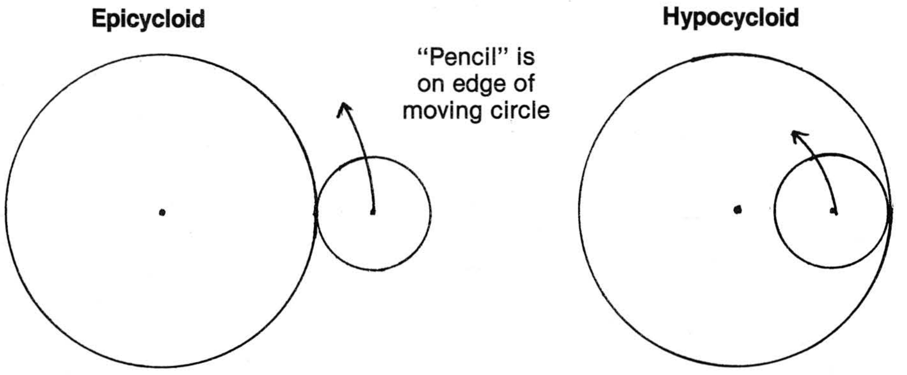
Your screen probably will be different from mine. Do this. In Line 4 make ASPECT = 1. In line 5 make A = 100000. In line 6 make B = 1. This will attempt to draw a circle on your screen. If it doesn't come out a circle, get a ruler. Divide the width of the image by the height. In line 3, type "SCIRCLE =" [the value you just calculated]. In line 4, type "ASPECT = SCIRCLE." RUN the program again, and this time you should have a circle.

If you have a printer, again set ASPECT = 1, RUN the program, and dump the resulting image to paper. Divide the width of the output by the height, and in line 3, add the statement "PCIRCLE = " [your calculated value]. In line 4, you can now set "ASPECT = PCIRCLE" when you want circular printouts, and "ASPECT = SCIRCLE" when you want circular screen images. Obviously, "ASPECT =" [anything else] will give you ovals. Different monitors and printers will require different aspect ratios, because of the way they space their dots.

What about patterns? The pattern you get depends upon the relative sizes of the "fixed" and "moving" circles. The variable "A" is given the diameter of the fixed circle, and "B" the diameter of the movable circle. Though I don't know how to prove it, my observation is that, adjusting the fraction A/B to the lowest integer values for both A and B, the number of points in the pattern will be A, and the number of revolutions required to complete the pattern will be B. Try different combinations and see what happens.

```
1  REM  CYCLOIDS BY DAVID SPARKS

2  TEXT : HOME : REM
  CLEAR THE SCREEN
3 PCIRCLE = 176.4 / 134:SCIRCLE =
     6.875 / 5.4375: REM
  ABOVE ARE ADJUSTMENTS FOR
  PRINTER AND SCREEN ASPECT
  RATIOS ON MY SYSTEM
4 ASPECT = SCIRCLE: REM
  SET TO DRAW CIRCULAR
  IMAGES ON SCREEN
5 A = 8: REM
  RADIUS OF "FIXED" CIRCLE
6 B = 3: REM
  RADIUS OF "MOVING" CIRCLE
7 EPICYCLOID = 1: REM
  MAKE THIS NON-ZERO IF
  YOU DO WANT AN EPICYCLOID,
  ZERO IF YOU DO NOT.
8 HYPOCYCLOID = 0: REM
  MAKE THIS NON-ZERO IF
  YOU DO WANT A HYPOCYCLOID,
  ZERO IF YOU DO NOT.
9  REM
  MAKE BOTH EPICYCLOID AND
  HYPOCYCLOID NON-ZERO
  FOR A COMBINATION PATTERN

10  DEF  FN XH(Q) = ((A - B) *  COS
     (Q) + B *  COS (Q * ((A - B)
      / B)) - XL) / (XR - XL) * 2
     79: REM   X FOR HYPOCYCLOID
15  DEF  FN XE(Q) = ((A + B) *  COS
     (Q) - B *  COS (Q * ((A + B)
      / B)) - XL) / (XR - XL) * 2
     79: REM   X FOR EPICYCLOID

20  DEF  FN YH(Q) = ((A - B) *  SIN
     (Q) - B *  SIN (Q * ((A - B)
      / B)) - YT) / (YB - YT) * 1
     91: REM Y FOR HYPOCYCLOID
25  DEF  FN YE(Q) = ((A + B) *  SIN
     (Q) - B *  SIN (Q * ((A + B)
      / B)) - YT) / (YB - YT) * 1
     91: REM Y FOR EPICYCLOID

30  IF EPICYCLOID THEN 100: REM
  BRANCH IF EPICYCLOID WANTED.
35  IF HYPOCYCLOID THEN 500: REM

   BRANCH IF HYPOCYCLOID WANTED
40  PRINT "YOU DIDN'T SELECT EITH
     ER PATTERN.": PRINT  CHR$ (7
     ): END : REM
  RING THE BELL AND STOP IF
  NEITHER PATTERN SELECTED.

100  REM      EPICYCLOID

110  HGR : POKE  - 16302,0:X = B *
     2 + A: REM
SELECT AND CLEAR HI-RES SCREEN,
SELECT FULL-SCREEN DISPLAY,
CALCULATE SCALING PROPORTION
FOR EPICYCLOID.

115 XL =  - X * ASPECT:XR = X * A
     SPECT:YB =  - X:YT = X: REM
     SCALING PARAMETERS

120  HCOLOR= 3: HPLOT  FN XE(0), FN
     YE(0): REM
     SELECT WHITE AS COLOR AND
     PLOT STARTING POINT.
```

```
130 :: FOR Q = 0 TO B * 8 *  ATN
     (1) STEP 8 *  ATN (1) / 360:
      REM PROPER NUMBER OF REVO-
LUTIONS ONE DEGREE AT A TIME

140  HPLOT  TO  FN XE(Q), FN YE(Q
     ): REM NOTE HOW A CALL
TO A DEFINED FUNCTION CAN BE USED
     ANYWHERE A REGULAR NUMERIC
CONSTANT OR VARIABLE IS ALLOWED.

150  NEXT
200 ::: IF HYPOCYCLOID THEN 520::
      REM IF BOTH EPICYCLOID AND
HYPOCYCLOID DESIRED, THEN BRANCH
INTO HYPOCYCLOID MODULE PAST ITS
INITIALIZING-SCALING INSTRUCTIONS

210  GOTO 700: REM

500  REM         HYPOCYCLOID

510  HGR : POKE  - 16302,0
512 X = (A * (A >  = B)) + ((2 *
     B - A) * (B > A)): REM  USE
OF BOOLEAN LOGIC TO DETERMINE
DIMENSIONS OF FIGURE
515 XL =  - X * ASPECT:XR = X * A
     SPECT:YB =  - X:YT = X: REM
     SCALING PARAMETERS
520  HCOLOR= 3: HPLOT  FN XH(0), FN
     YH(0)
530  FOR Q = 0 TO B * 8 *  ATN (1
     ) STEP 8 *  ATN (1) / 360
540  HPLOT  TO  FN XH(Q), FN YH(Q
     )
550  NEXT
590  GOTO 700: REM

700  REM         FIND OUT WHETHER
      PRINTOUT DESIRED

710  POKE  - 16301,0: REM SELECT
     MIXED TEXT AND GRAPHICS
720  VTAB 20: PRINT : REM MOVE TO
      TEXT WINDOW ON SCREEN
730  INPUT "DO YOU WANT A PRINTOU
     T? ";X$
735  POKE  - 16302,0: REM GO BACK
      TO FULL-SCREEN DISPLAY
740  IF  LEFT$ (X$,1) <  > "Y" THEN
     2000: REM
    GO TO PROGRAM ENDING SEQUENCE
      IF NO PRINTOUT DESIRED.

1000  REM      PRINT THE PICTURE

1003  REM
PRINTER OUTPUT NOT REQUIRED TO
RUN PROGRAM. SEE YOUR OWN SYSTEM
MANUALS OR DEALER FOR HI-RES
PRINTING INSTRUCTIONS ON YOUR
SYSTEM, AND INSTALL CODE
AT LINE 1010
1010  REM (HI-RES DUMP CODE HERE)
1020  PRINT  CHR$ (4);"PR#1": FOR
     Q = 1 TO 33: PRINT : NEXT : PRINT
      CHR$ (4);"PR#0": REM  PRINT
      ENOUGH BLANK LINES TO EJECT
      SHEET FROM PRINTER.
1030  GOTO 2000: REM

2000  REM PROGRAM END SEQUENCE
2010  END : REM
      MORE COULD BE DONE HERE,
    BUT WE'LL LEAVE THAT TO YOU.
```



EPICYCLOID and HYPOCYCLOID, in the program, are *boolean* variables, or true–false tests. Any Applesoft numeric variable can be used as a boolean. Applesoft defines the value zero as "false," and any non–zero value as "true." This program uses non–zero, or "true" values to mean "draw one of these."

In the program as printed here, the variable EPICYCLOID is given a value (1) in line 7, and the variable HYPOCYCLOID is given the value zero in line 8. Line 30 begins, "IF EPICYCLOID . . ." Notice it doesn't say, "IF EPICYCLOID = 1. . ." This is an example of boolean, or true–false use of a variable. With EPICYCLOID = 1, the program will "branch" to line 100. Try changing the values in lines 7–8 and see if you can follow the boolean logic in lines 30, 35, and 200.

Epicycloids are curves "described" by a point on the circumference of a circle rolling freely, but without any slippage around the outside of the circumference of a second, fixed circle. Hypocycloids are a similar concept, except the rolling circle moves inside the circumference of the fixed circle:

Here are the Sussmans' *Parametric Equations* for the two types of curves:

Epicycloid
$x = (a+b)\cos(Q) - b\cos(gQ)$
$y = (a+b)\sin(Q) - b\sin(gQ)$

Hypocycloid
$x = (a-b)\cos(Q) + b\cos(hQ)$
$y = (a-b)\sin(Q) - b\cos(hQ)$

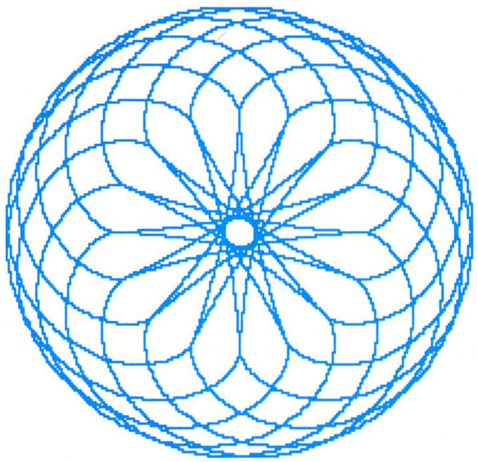
where
a = radius of fixed circle
b = radius of rolling circle
Q = angle between the x-axis and the line connecting the two circles' centers, in radians
$g = (a+b) \div b$
$h = (a-b) \div b$

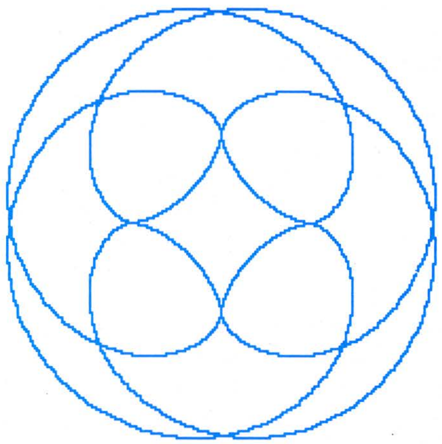
Well, time passes, and children grow up. Cousin Ricky is now Richard, a hot–shot computer geology expert for a big oil company. He makes computers print 3–D color pictures of rocks three miles down. But I, I can produce Spirographs of many shapes. And this time, the toy belongs to me.

I've had a lot of fun with this. A young lady in my household loves to color the printouts. I've printed them on tracing paper, then used my darkroom enlarger to "blow up" the image onto the top of a cake. We traced the image in icing and had a wonderful time. It's fascinating to just sit and watch them go. And I've only used **HCOLOR=3**, which is white. I've yet to try changing colors.

If you select both an epicycloid and a hypocycloid, the screen will be scaled for an epicycloid, the pattern will be drawn, then the hypocycloid will be produced inside it. With a bit of work, you might be able to get multiple, concentric patterns by proportionally adjusting both A and B.

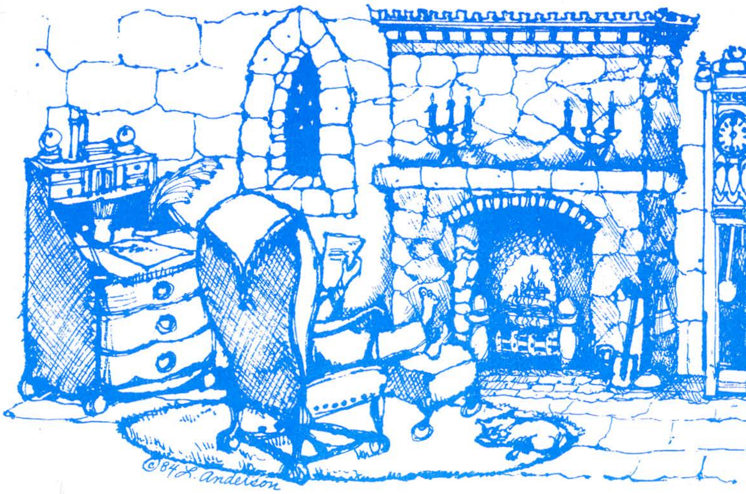
As written, the program runs on an Apple][Plus, 64K, Epson MX–80 printer with Graphtrax 80, and a Mountain Hardware CPS Multifunction Card. It should run equally well on a 48K Apple. Graphics printout is obtained with The Routine Machine and Amperchart from Roger Wagner Publishing. Obviously, many graphics printer interface cards, such as the Dumpling GX, afford the ability to output Hi–Res images. There are also other software packages which can do the job.





The Duncan Letters

Joe Holt/Laurie Wofford



Duncan my Friend!

Salutations again from Fairhill Castle! The Winter in all her fury has finally left us, leaving behind a white world, which is just now beginning to turn green. Soon, the King's horses will be running wild on the East Pastures outside these stone castle walls, happy to finally see the spring.

And they aren't the only ones! The warmth of a fire is one thing, but the warmth of the sun . . . ah! I think I'll go out and find me a nice spot upon the walkways around the top of the inner wall, then continue this letter . . .

Much better! After seeing the four walls of my quarters for so long, and only white outside my window, it's refreshing to gaze westward and see the familiar forests of Greenwood, stretching even as far as the Far Mountains. I can almost imagine your small village on the other side of those snowy peaks . . .

Which reminds me: When I'd sent you my last letter, and some weeks after had still not heard a word from you, I began to worry. Had the nomadic Wanderers raided your village? I shuddered at the thought. I decided to pay the Sorceror and his mail service a visit down in his dismal hole under the castle to find out if my letter had gotten delivered safely to you.

Of course, when I arrived in the Sorceror's chamber, he was busy tending to some new experiment. He was hunched over a table, staring into a small upright box. A glow from inside the box illuminated the Sorceror's face. Its flickering cast made him look even stranger than he already does. His right hand was resting on another, much smaller box, sitting on the table next to the other.

I politely interrupted him, and after excusing myself numerous times (it is unwise to upset a sorceror), asked if the mail had been delivered over the Far Mountains and to your village safely. He stood up, and after setting down what he later told me was a "mouse," informed me that the bad winter had taken its toll on his Hexels who deliver the mail, but yes, the mail had gotten through, although a little behind schedule.

My mind thus relieved, I began to wonder what on earth was inside that box of his.

I ventured to ask. "Excuse me, Sorceror, but, well, if it won't be intruding too much, I was wondering, well, if — what's inside that box on your table?" His response took me by surprise, to say the least.

"I knew I couldn't keep it a secret for very long! Very well, then. There is Magic inside this box, Joseph. Inside this box is power. It is the world at your fingertips. Ah, but more than the world — it is anything you could ever imagine! This, Joseph, is a Magic Apple!"

A Magic Apple! One of the very kind which that traveling minstrel had spoken of! All the wonderful things the minstrel had claimed Apples could do, and how fantastic he'd claimed they were . . . and the Sorceror has had one all the time! I was taken aback, and I told him so.

He was upset. "What did you think I did my magic with? Newt's eyes and bat's wings?!?" Well, I must confess that your cousin's knowledge of magic is very limited . . . I was too embarassed to answer honestly.

"But, that doesn't look at **all** like an apple," I said cautiously. "It's square, and white, and **glows**!"

"Of course it doesn't look like an ordinary apple, because it isn't! You don't eat Magic Apples," the Sorceror replied.

"Thank goodness. I don't think that **that** apple would go down very easily. In fact, it doesn't look very edible at all." I stepped closer to the Magic Apple, and stared at it for a while. There were images on one side of the box, Duncan, glowing images! It was like bright candles placed behind a sheet of writing paper which someone had scribbled upon.

Then the sorceror spoke: "Well . . . go ahead."

"You mean I can touch it?" I asked.

"Of course you can **touch** it! I mean, you can take it back to your room and use it for a while. It's not difficult, and I'm sure you can quickly grasp it."

I was more than shocked by this unusual show of generosity. Think of it, Duncan: A Magic Apple! I would be able to use one of these Magic Apples myself!

The sorceror then explained that he had many other types of apples in his cubby hole to keep him busy for a while, and that he was anxious to see how easy it would be to learn for someone who'd never used one of these apples before. I suppose he picked a good person for that!

He talked at length about this particular Magic Apple, which he called a "Macintosh." He said that the wizard Woz, whom I told you about in the last letter, had originally created the Apple I, which was the first of the Magic Apples. Then, not content with giving the world one Magic Apple, he created another, named (cleverly enough) the Apple II. After a little help from a friend, he began to sell these Magic Apples, and soon they were in every imaginable marketplace.

Eventually, an entire proprietorship was formed because of the success of these Magic Apples. The sorceror said that his Hexels which got the apples from Silicon Valley told him that it is almost like a small city, peopled by an entire band of Wizards working together to create even better Magic Apples!

And that's what this Macintosh Magic Apple is. Although the wizard Woz had little to do with this particular apple, the sorceror said that it indeed is as wonderful as the original apples.

So, Duncan, that's how Macintoshes came about, and how I have in my room right now one of these Magic Apples.

After the sorceror explained all this, he carried the Macintosh up to my room (in a basket, so none of the other inhabitants here would see it; he doesn't want word of this to get out right now. I agree; I want to have it all to myself for a little while, at least!). He set it on my drawing table and started looking around on the walls for something. He said he was looking for an "electrical outlet." Did I ever tell you about these?

Well, I've always been suspicious of all that copper tubing the Sorceror ran throughout the castle last summer, even to strange places like the dungeon and the tower. Ever since he fell into the moat, I think his brain has been a little waterlogged. I recall I've told you about his peculiarities, have I not? Flying kites in thunderstorms, and muttering things about "electricity" and such (that pointed hat of his is surely cutting off the circulation to some vital area).

Anyway, this copper tubing, through which he claims his "electricity" flows, is used to fuel the Macintosh, or so he says. Between you and me, Duncan, I've placed my ear against the tubing many times, and I hear nothing flowing! But it's true. When I connect the Sorceror's Apple to the tubing, it tells me all sorts of things (some of which I don't yet understand, like "VIEW BY ICON"), but when I "pull the juice" (as the Sorceror is fond of saying), which is to say that I disconnect the Apple from the tubing, it suddenly goes quiet and the glow fades, and even my most urgent promptings are ignored by it.

The Sorceror has this to say about electricity and Magic Apples:

"Apples use electricity as power, and, through a most ingenious internal switching power supply, are able to transform 120 volt AC castle current to various DC voltages, such as +5, −12, ground, and so on, which, when applied to the various components within the Apple, integrated circuits being the majority . . . er, Joseph, where was I?"

What I think he's trying to say is that the electricity is like food for the Macintosh, just like ordinary apples are food for us, and without food, the Apple will die. This sounds all very simple now that I think back upon it, and even though the Sorceror has told me that Apples will "come back to life" when re-supplied with their electric food, I've yet to turn it off. We've only just been introduced, but I already feel as if the Macintosh is my friend. You can't just go around disconnecting your friends!

This is what I've been able to figure out about how Magic Apples are able to help me with my humble goings-on:

First, of course, is the fact that Apples simply cannot do anything without being connected to electricity. Try as you might, they just won't.

Second, Apples need to be told what to do, whether by me or by someone else. Usually when I "talk" to the Apple, I'm actually talking to it through the instructions of someone else. Hmmmm . . . now I'm starting to sound like the Sorceror . . .

That second thought is somewhat difficult to understand. I've got an idea which may explain it a little better:

Do this: next time your parents ask you to feed the unicorn, shout at the top of you lungs "I CERTAINLY WILL!" And next time they ask you if you've done your studies, spin three times in a circle and tell them (closing one eye) "YES I HAVE!" And before they even ask if you've washed behind your ears, when you see that "have you washed behind your ears" look on their faces, hop on one foot and whistle, then hop on the other foot and say "OF COURSE!".

Have you got that? Let's give it a try . . .

"Duncan my son, will you feed the unicorn?"
[you say:]
"I CERTAINLY WILL!"
"Duncan dearest, have you done your studies?"
[spin three times, close one eye:]
"YES I HAVE!"
"Duncan, oh Duncan, have you . . ."
[hop on one foot, whistle, hop on the other:]
"OF COURSE!"

And then if your parents ask you why you're acting so strangely, tell them your cousin Joseph told you to (then, when your parents look at you sideways, you'll know better than to do everything your cousin Joseph tells you to do!).

You see, just as I told you what to do when your parents ask you certain things, so has the wizard Woz and his friends told their Apples what to do if I ask it things. The Macintosh is unique, according to the sorceror, in that to use it, you hardly have to know anything about these instructions the Wizards gave it. Thank goodness for that!

Of course, Magic Apples don't have ears to wash behind or unicorns to feed, but they can do many useful things (or so the Sorceror tells me). All these things will have to wait until next time, because my hand has grown weary, and my mind is soon to follow. And so, Duncan, to tell you the truth, I want to go back to my quarters and play with the Macintosh!

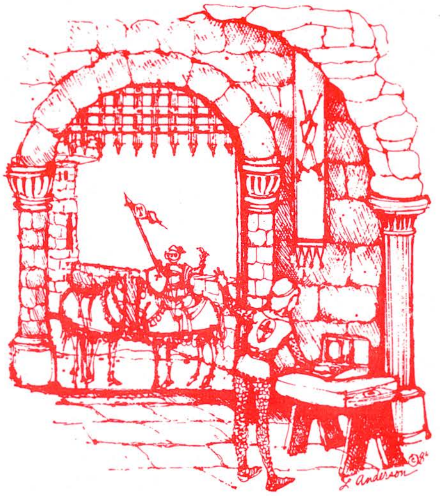
It's very hard, not being able to tell anyone else

Continued on Page 26

Assembly Language for Teen-agers

The Sourceror's Apprentice

Mike Newton



Welcome back to Castle Fairhill, my apprentice! I know that your father, Sir Touhee, waits outside with your mount, to take you to the Tuldurry Knights Tournament, so we'll get right to your studies.

Wizards have all kinds of important tools and helpful items. Where would a wizard be without his staff? Up the stream without an oar, I'll warrant. There are a lot of other magic tools as well, like cloaks of invisibility, eyes of Newt, seeing stones, ancient scrolls of enchantments. . .well, you get the idea. Working magic without one's tools is extremely difficult, if not impossible.

Just as wizards have these special tools, programmers have many programming tools, commonly called *utilities*. Your editor/assembler is one of these tools, and the most important one to an assembly language programmer. Other tools include base conversion charts, debuggers, reference manuals and monitors. Life can be made much simpler with these programming utilities.

Every Apple computer contains a built-in *monitor*, hidden away amongst the Apple ROMs. Don't confuse this monitor with the *video monitor* that displays text and graphics and looks like a TV set. The monitor utility program allows you to "get into" the Apple memory. It has commands to store values in memory, inspect the values in memory, jump to a program in memory, and many other useful functions.

Let's get acquainted with the monitor. From Applesoft, type "CALL -151" and then press the RETURN key. You should now see an asterisk (*) at the far left side of the screen and a blinking cursor. The asterisk is the monitor *prompt*, and always indicates that you have "entered" the monitor. A prompt is a special character that is used to tell you which of the Apple's operating systems you are currently using.

The first monitor command that we will learn about is the "G" command. The "G" stands for "Go," and is more or less the machine language equivalent of the Applesoft command "GOTO." The monitor commands always come after the data that the command operates on or uses.

Among the subroutines in the Apple monitor ROM is a routine called "BELL." As you might guess, this routine beeps the speaker with the same annoying sound that occurs when your Applesoft program makes a ?SYNTAX ERROR. The BELL routine begins at memory location $FF3A — notice that this is a hexadecimal number (all numbers and addresses used in the monitor must be expressed in hexadecimal). To call this routine, and thus beep the speaker, you must type "FF3AG" from the monitor and then press the RETURN key. The monitor will then "GOTO" to FF3A, where the BELL routine starts. The BELL routine beeps the speaker and then returns to the monitor. If you're wondering why we suddenly dropped the dollar sign ($) in front of the hex numbers above, it is because when we are working in assembly language, all numbers and addresses are assumed to be hexadecimal.

Well, the BELL routine is none too exciting. In fact, individually, none of the routines in the monitor are very exciting, but then a lot of programming is sort of drab. However, taken as a whole, the Apple monitor is as fine a collection of utility routines as you might hope for. While they are all we have to work with for a while, there's power in them thar bytes. Live without excitement for a while longer, then see how using these unexiting subroutines can make things easy for you.

Let's call another unexciting monitor routine, "ERR." This routine, located at $FF2D, prints the word "ERR" on the screen and then calls the BELL routine, which of course beeps the speaker. To call the ERR routine, we enter "FF2DG" into the monitor.

Let's call just one more routine from the monitor ROM. This routine is the first one that you see when you turn on your Apple — that's right, this routine clears the screen, beeps the speaker and prints the word "Apple] [" at the top of the screen. It is located at memory address $FB60. To call it, we simply enter "FB60G," which says to the monitor, "go to the machine language routine starting at memory location FB60".

Let's learn another monitor command, the List command. As you might guess, this command is very much like the Applesoft LIST command. Instead of listing an Applesoft program, however, the List command *disassembles* a machine language program in memory. Disassembling is, of course, the exact opposite of assembling: a disassembler converts machine language binary numbers into readable assembly language *mnemonics*.

A mnemonic is a made-up word that is easy to memorize, because the characters stand for something we can understand. For example, the mnemonic "LDA" means **L**oa**D** the **A**ccumulator. The monitor includes a disassembler which disassembles the raw binary data in memory, and converts it into mnemonic *opcodes* and *operands*. Opcodes and operands are terms we will discuss a little later on.

Even though we haven't started learning the many assembly language mnemonics yet, let's use the List command to disassemble memory anyway, just for the practice. We'll disassemble the BELL routine, and see what the machine language commands to beep the speaker look like. To do this, we simply enter the address and follow it by the monitor command: "FF3AL".

You should see all kinds of letters and numbers on the screen that probably don't mean much to you. I'll briefly explain what these different things mean. At the far left of the screen are hexadecimal addresses, ranging from FF3A to FF59. The next set of numbers are the bytes that comprise the current machine language instruction. Instructions are made of from one to three bytes, depending on exactly what the instruction is. Following this group of machine language numbers is the disassembled assembly language *instruction*.

The first disassembled line should look like this:

```
FF3A—     A9 87     LDA #$87
```

This line means: starting at hexadecimal address $FF3A are the bytes $A9 and 87, which comprise the machine language instruction which is "LDA #$87" in assembly language. The "LDA" is the mnemonic *opcode*, and when converted to machine language, tells the computer *what* to do. The "#$87" is the *operand*. Once the computer knows what to do, it must be told what to do it *with*. This is the function of the operand. As an assembly language programmer, you'll probably use the disassembling feature of the Apple monitor quite a bit in debugging programs that have gone haywire.

Another monitor command that is used often is the command to store bytes in memory. Let's say, for instance, that we wished to store the value $11 at memory location $2000. To tell the monitor to do this, we would simply enter "2000: 11 [RETURN]." If we wished to store a series of numbers starting at an address, you merely specify the starting address, type the colon (:), and enter the series of hexadecimal numbers. Remember that each hexadecimal number must consist of two hexadecimal digits. The value $4, for instance, *must* be entered as 04. If we wished to store the values $AB C2 D1 1F starting at the address $3000, for instance, the monitor entry would be "3000: AB C2 D1 5F [RETURN]." Note that there is a space between each hexadecimal number.

If you wish to see the values of bytes stored in memory, you can use the dot (.) command. If you wished to inspect memory from $2000 to $20FF, for instance, you would enter "2000. 20FF [RETURN]." While this is a very simple command, it is used often to print the contents of memory. To inspect the contents of a single memory location, you can just enter the address of the location to be inspected. The monitor will reprint the hexadecimal address and follow it with the hexadecimal contents of that location.

Now that we've learned all these new commands, let's do something with them. Look at Table I. (An enlarged copy with additional details, and printed on cardstock, is available. An ad appears elsewhere in this issue.) Table I is a chart of the values of characters on the screen. A normal "A", one that is not inverse or flashing, for instance, has a value of $C1. The text screen is an area of memory which starts at $400 and ends at $7FF. Let's put some characters on the text screen. To do this, we can use the monitor colon (:) command to store a series of values on the screen.

From Applesoft, type "HOME [RETURN]" to clear the screen. Then type "CALL -151 [RETURN]" to enter the monitor. The screen display memory starts at address $400, and if we begin storing values there, they will appear on the top of the screen.

Let's put the word "MAGIC" on the screen. Referring to the ASCII code chart, we can see that this would be the values $CD C1 C7 C9 C3. To "PRINT" the word on the screen, we can type in the monitor command "400: CD C1 C7 C9 C3 [RETURN]," and as if by magic, the word "MAGIC" should now be at the top of the screen.

Have you ever wondered how the Applesoft "PRINT" command works? Well, now you know. It simply stores ASCII values to the memory area that is associated with the screen text display. That's all that most machine language programs do: store various values at various memory locations.

An understanding of the Apple][monitor is very important for an assembly language programmer. It is usually the only thing that there is to help you debug your machine language. There are all sorts of assembly/machine language aids available. The only tools that we will use together, however, are the monitor and the editor/assembler.

There are so many things that you need to know before you can program in assembly language that it has taken us a while to get as far as we have gotten. My master always told me that a journey of a thousand miles starts with a single step, and that is what we've been doing: taking those first short and simple steps. Being an assembly language wizard is truly an infinite journey: there is always a new exciting landscape on the horizon, some new unexplored land. Even an old wizard is always learning new things.

I bet you're really looking forward to the Tuldurry Knights Tournament. I remember seeing it as a boy. That iron mail shirt you're wearing, isn't that the official coat of mail of the Tournament? It's really rather strange how those Knights are boycotting the Tournament, isn't it? But what else should this year's host expect after boycotting the last Tournament himself? Aren't people strange, especially those Knight types? Always looking for something to quibble over! Have fun, and come back soon!

	Inverse				Flashing				Normal							
									Ctrl						L/case	
	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
$0	@	P		0	@	P		0	@	P		0	@	P	'	p
$1	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	a	q
$2	B	R	"	2	B	R	"	2	B	R	"	2	B	R	b	r
$3	C	S	#	3	C	S	#	3	C	S	#	3	C	S	c	s
$4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	d	t
$5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	e	u
$6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	f	v
$7	G	W	'	7	G	W	'	7	G	W	'	7	G	W	g	w
$8	H	X	(	8	H	X	(	8	H	X	(	8	H	X	h	x
$9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	i	y
$A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	j	z
$B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	k	{
$C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	l	¦
$D	M	]	-	=	M	]	-	=	M	]	-	=	M	]	m	}
$E	N	^	.	>	N	^	.	>	N	^	.	>	N	^	n	~
$F	O	_	/	?	O	_	/	?	O	_	/	?	O	_	o	■

Table I: ASCII Screen Characters

Apple® Programming Utilities

☐ GPLE™ (Supports DOS 3.3 *and* ProDOS™)

GLOBAL PROGRAM LINE EDITOR by NEIL KONZEN
$49.95: Includes Peeks/Pokes Chart & Tip Book #7

THE NUMBER 1 APPLE PROGRAM LINE EDITOR GPLE lets you edit 40- or 80-column Applesoft program lines *FAST* without awkward cursor-tracing or "Escape editing". Compatible with Double-Take's 2 way scrolling.

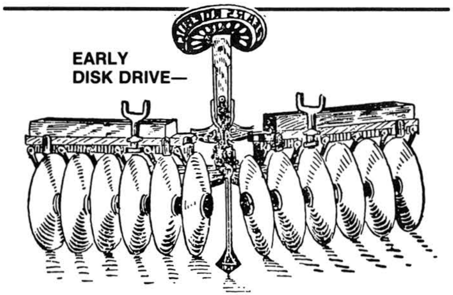
INSERT & DELETE: GPLE works like an in-memory word processor for Applesoft program lines. Simply jump the cursor to the change-point and insert or delete text. No need to trace to the end of a line before hitting Return.

GLOBAL SEARCH & REPLACE: Find any word or variable in your programs, *FAST*. For example, find all lines containing a GOSUB, or all occurrences of variable XY. REPLACE ANY VARIABLE or word with any other. For example, change all X's to ABC's, or all "Horses" to "Cows".

DEFINABLE ESC FUNCTIONS: Define ESC plus any key to perform any task. For example, **ESC-1** can catalog drive 1, or **ESC-N** could type an entire phrase or subroutine. *Anything* you want, *whenever* you want.

GPLE DOS MOVER: Move DOS 3.3 & GPLE above main memory for an EXTRA **10K** of programming space.

PLUS APPLE TIP BOOK #7: Learn more about your Apple! Includes all-new useful GPLE tips and tricks.



☐ BEAGLE BASIC™ (REQ. 64K)

APPLESOFT ENHANCER by MARK SIMONSEN
$34.95: Includes Peeks/Pokes Chart & Tip Book #6

RENAME ANY APPLESOFT COMMAND or Error Message to anything you want. For program clarification, encryption/protection or even foreign translation. Example:

```
10 POUR X=1 TO 3: ECRIVEZ "BONJOUR": ENSUITE
```

RAM Applesoft is *better* Applesoft! Beagle Basic replaces those obsolete cassette commands (SHLOAD, etc.), with powerful new commands that you can USE—

ELSE follows Applesoft If-Then statements, like this:

```
IF X=2 THEN PRINT "YES": ELSE PRINT "NO"
```

HSCRN reads the color of a hi-res dot for collision testing, **SWAP** exchanges variable values, **TONE** writes music without messy Pokes or Calls, **SCRL** scrolls text in *either* direction, **TXT2** lets Text Page 2 act exactly like Page 1...

GOTO AND GOSUB may precede variables, as in "GOSUB FIX" or "GOTO 4+X". Escape-mode indicated by a special ESCAPE CURSOR. Replace those awkward Graphics screen-switch pokes with one-word commands. Change your ctrl-G Beep to any tone you want. **INVERSE REM STATEMENTS** too! GPLE/Double-Take compatible.

☐ DOS BOSS™

DOS 3.3 EDITOR by BERT KERSEY & JACK CASSIDY
$24.00: Includes Peeks/Pokes Chart & Tip Book #2

RENAME DOS COMMANDS and Error Messages—DOS 3.3's "Catalog" can be "Cat"; DOS's cryptic "Syntax Error" can be "Oops" or almost *anything* you want it to be.

PROTECT YOUR PROGRAMS. Unauthorized Save-attempts can produce "Not Copyable" message, or *any* message. **List-Prevention** and other useful Apple tips and tricks. Plus one-key program-execution from catalog.

CUSTOMIZE DOS. Change Disk Volume headings to your message or title. Omit or alter catalog file codes. Fascinating documentation, tips & educational experiments.

ANYONE USING YOUR DISKS (booted or not) will be forced to use DOS formatted the way YOU designed it.

☐ DOUBLE-TAKE™ (DOS 3.3 *and* ProDOS™)

2-WAY SCROLL/MULTI-UTILITY by MARK SIMONSEN
$34.95: Includes Peeks/Pokes Chart & Tips Chart #1

2-WAY SCROLLING: Listings & Disk Catalogs scroll Up *and* Down, making file names and program lines faster to find and easier to access. Change Catalog or List scroll-direction with Apple's Arrow keys. Machine Language and Hex/Ascii dumps scroll two-ways too. All features are GPLE compatible and support 80-column display.

BETTER LIST FORMAT: Each Applesoft program statement lists on a new line for *FAST* program tracing & de-bugging (see sample below). Printer-compatible in any column-width—Great for archive printouts.

```
A$="DOGFOOD"
X=3.14159
Y=255
```

← **VARIABLE-DISPLAY:** prints all of a program's strings and variables with their current values.

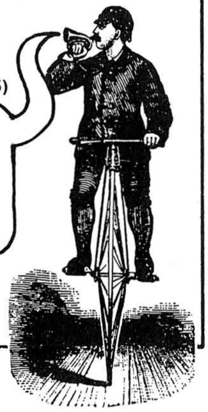
```
A$: 100 200 250
X: 10 20 3000
Y: 10 40 55 60
```

← **CROSS-REFERENCE:** Sorts & displays line numbers where each variable & string appears.

AUTO-LINE-NUMBER, instant Hex/Dec Converter, better Renumber/Append, Program Stats, Eliminate/Redefine Cursor, Free Space-On-Disk... All GPLE/Pronto compatible.

2-WAY VIDEO SCROLLING

```
]LIST
10 HGR2
  : FOR Y=0 TO 191
  : POKE 228, C
  : C=C+1/9-256*(C=255)
20 REM "Double-Take
   optionally lists each
   Applesoft program
   statement on a new
   line, making pro-
   gram code much
   easier to read."
30 HPLOT 0,Y TO 279,Y
  : NEXT Y
  : POKE 2053,58
  : GOTO 10
```



High-Speed DOS

Multi-Utilities

☐ PRONTO-DOS™

HIGH-SPEED DOS/DOS-MOVER by TOM WEISHAAR
$29.50: Includes Peeks & Pokes Chart

TRIPLES THE SPEED of disk access and frees 10,000 bytes of extra programming memory by moving DOS 3.3.

Function	Normal	Pronto
BLOAD HI-RES IMAGE	10 sec.	3 sec.
LOAD 60-SECTOR PROGRAM	16 sec.	4 sec.
SAVE 60-SECTOR PROGRAM	24 sec.	9 sec.

(Text-files no change) **Bload language cards** at *triple speed.* Create bootable high-speed disks with the normal INIT command. Compatible with *all* commands, GPLE, Double-Take, DOS Boss, DiskQuik & most unprotected programs.

MOVE DOS 3.3 above main memory to free *10,000 bytes* of memory for your programs (64K required to move DOS).

15 EXTRA SECTORS per disk. Catalog Free-Space is displayed on the screen every time you Catalog a disk.

NEW TYPE-COMMAND ("TYPE filename") prints the contents of any Text File on-screen or to your printer.

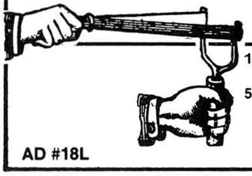
☐ DISKQUIK™ (Requires IIe with 128K)

DISK EMULATOR by HARRY BRUCE & GENE HITE
$29.50: Includes Peeks & Pokes Chart
Requires Apple IIc or IIe with EXTENDED 80-col. card)

ACTS LIKE A DISK DRIVE in Slot 3, but super-fast and silent! Enjoy many of the benefits of another drive at 1/10th the cost. Catalog with "CATALOG, S3" command. Access all kind of files in RAM with normal DOS commands.

SILENT AND FAST: Since no moving parts are involved, DiskQuik operates at super-high speeds. See to believe! Your Apple IIe's Extended 80-column Card (required) holds about half the amount of data as a 5¼" floppy!

MANY USES: For example, load often-used files like FID into RAM when you boot up, so they are always available when you need them. Copy files from RAM onto disk and vice versa, just as if a disk drive were connected to slot #3.



COMPATIBLE with all normal DOS procedures.

```
1234 TEXT: HOME: NORMAL:
     PRINT CHR$(21)
5678 R=INT(RND(1)*10): N(R)=
     N(R)+1: VTAB R+9: HTAB
     40: PRINT CHR$ (124); SPC
     (N(R)); CHR$(R+65); :
     IF PEEK(36) THEN 5678
```

AD #18L

☐ UTILITY CITY™

21 PROGRAMMING UTILITIES by BERT KERSEY
$29.50: Includes Peeks/Pokes Chart & Tip Book #3

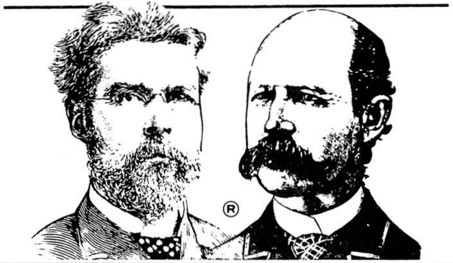
LIST FORMATTER prints each Applesoft program statement on a new line. For-Next Loops are indented with printer Page Breaks. A great Applesoft program de-bugger.

MULTI-COLUMN CATALOGS to your printer, with or without sector and file codes. Organize your disk library.

INVISIBLE AND TRICK catalog File Names. Put invisible functioning commands in Applesoft programs too.

21 UTILITIES TOTAL, including auto-post Run-number & Date in programs, alphabetize/store info on disk, convert dec to hex or Int to FP, protect and append programs, dump 40-column text to printer. And More.

LEARN PROGRAMMING TRICKS: LIST-able programs and informative documentation. Includes Tip Book #3— Hours of good reading and Applesoft experiments.

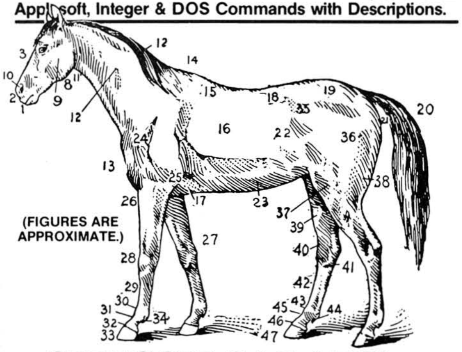


Beagle Bros
Micro Software Inc.

3990 OLD TOWN AVENUE, SUITE 102C
SAN DIEGO, CA 92110 / 619-296-6400

Attention Applers: Most Apple dealers and software stores have Beagle Bros products on their shelves. If you can't find the disk you want, bug the manager—he can have *any* of our products in his store for you within a couple of days.

FREE APPLE COMMAND CHART: Each SILICON SALAD and TIP DISK #1 comes with an 11x17 poster of all Applesoft, Integer & DOS Commands with Descriptions.



(FIGURES ARE APPROXIMATE.)

☐ SILICON SALAD™

WITH TIP DISK #2 by BERT KERSEY & MARK SIMONSEN
$24.95: With Peeks/Pokes Chart & Apple Command Chart

MANY MINI-UTILITIES: Program Splitter makes room for hi-res pix in large Applesoft programs, **Disk Scanner** finds bad disk sectors, **Key-Clicker** adds subtle sound as you type, **DOS-Killer** adds two tracks of space to your disks, **2-Track Cat** allows up to 210 DOS 3.3 file names per disk, **Text Imprinter** converts text-screen text into hi-res text, **Onerr Tell Me** prints the appropriate error message but continues program execution, **Text Screen Formatter** formats and converts text layouts into Print statements... plus much more Apple wizardry from the boys at Beagle Bros.

MORE TIPS ON DISK: Over 100 programs from Beagle Bros Tip Books 5, 6 and 7; *and* from Tip Chart #1.

TWO-LINERS TOO: From our customers around the world—and elsewhere. New tricks for your old Apple!

☐ TIP DISK #1™

100 TIP BOOK PROGRAMS ON DISK by BERT KERSEY
$20.00: With Peeks/Pokes Chart & Apple Command Chart

100 LISTABLE PROGRAMS from Beagle Tip Books 1-4. Make your Apple do things it's never done! All 100 programs are LISTable and changeable for experimentation. Two-Liners too, plus a free *Apple Command Chart.*

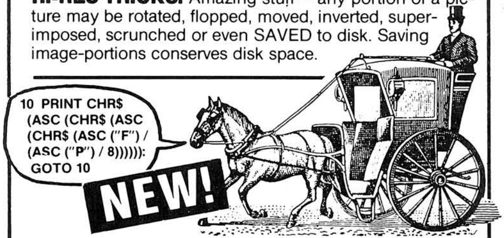
NEW!

Apple® Graphics Utilities

☐ BEAGLE GRAPHICS™

DOUBLE HI-RES GRAPHICS by MARK SIMONSEN
$59.95: Includes Peeks & Pokes Chart
Requires Apple IIc (or IIe with EXTENDED 80-col. card).

560-WIDE WITH 16 COLORS! Beagle Graphics gives your 128K-Apple (required) a full range of colors plus *double* high resolution—560 x 192 pixels. All Applesoft hi-res functions, including shape tables, are supported.
NEW COMMANDS let you draw fast circles, ellipses and rectangles from the keyboard or from your programs.
FAST COLOR FILL fills any outline with one of 16 solid colors or 256 color mixes (usable in your programs).
PROGRAM & PICTURE-CONVERTERS change your existing Applesoft programs *and pictures* to double hi-res. "Double-ize" *Apple Mechanic* shape table programs too!
DOUBLE-PLOT PROGRAM lets you draw and manipulate pictures in double hi-res. Add type to pix too. Move sections of images to either page. Save pictures to disk.
HI-RES TRICKS: Amazing stuff— any portion of a picture may be rotated, flopped, moved, inverted, superimposed, scrunched or even SAVED to disk. Saving image-portions conserves disk space.



```
10 PRINT CHR$
(ASC (CHR$ (ASC
(CHR$ (ASC ("F") /
(ASC ("P") / 8)))))):
GOTO 10
```

☐ TRIPLE-DUMP™

SINGLE/DOUBLE HI-RES "PRINT-ANYTHING" UTILITY
$39.95: Includes Peeks & Pokes Chart

PRINT ANY SCREEN IMAGE on your dot-matrix (graphics-capable) printer— Hi-Res, Lo-Res, **Double Hi-Res**, Medium-Res, as well as 40 **and 80-Column Text**. All print functions may be used in your Applesoft programs and disks. Don't settle for a "locked up" printer-dump program.
SPECIAL EFFECTS: Crop, rotate, enlarge, distort, invert... Preview pictures on the screen before you print.
BANNER MAKER: Make impressive 8"-high signs and banners for your family, home or office. Type any message—no length limit—and let your printer do the work!

☐ ALPHA PLOT™

STANDARD HI-RES GRAPHICS UTILITY
by BERT KERSEY & JACK CASSIDY
$39.50: Includes Peeks/Pokes Chart & Tip Book #4
(Alpha Plot offers 6-color 280-pixel resolution and requires only 48K; see Beagle Graphics for 128K double hi-res.)

DRAW IN HI-RES on both pages using easy keyboard commands. Pre-view lines before plotting. Use solid or mixed colors and Reverse (background opposite). One-keystroke circles, boxes and ellipses, filled or not. All pictures are Save-able to disk for access by your Applesoft programs.
COMPRESS HI-RES DATA to 1/3 disk-space, allowing 3-times the number of hi-res pictures per disk (avg. figures).
MANIPULATE IMAGES: Superimpose pictures or *relocate* sections of images anywhere on either hi-res page.
HI-RES TYPE: Add variable-size color & b/w text to your pictures. Type *anywhere* with no htab/vtab limits. Type sideways too, for Charts & Graphs. Includes Tip Book #4.

☐ FLEX TYPE™

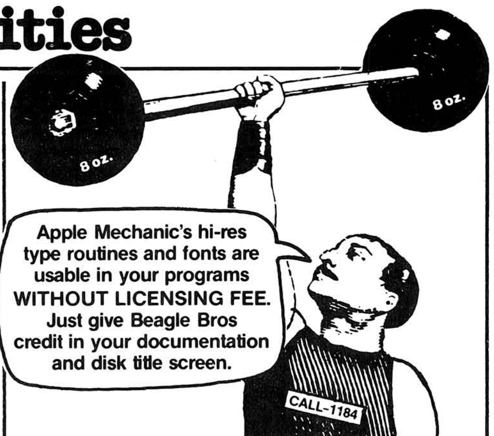
FLEXIBLE-TEXT UTILITY by MARK SIMONSEN
$29.50: Includes Peeks & Pokes Chart

PUT VARIABLE-WIDTH TEXT on both hi-res screens with normal Applesoft commands (including HTAB 1-70). Normal, expanded & compressed text with no extra hardware. (70-column text requires b/w monitor, not a tv).
COMBINE TEXT & GRAPHICS. Run existing Applesoft programs with Flex. GPLE/Double-Take compatible.
DOS TOOL KIT® font compatibility. Access up to 9 fonts in memory. Text editor lets you redesign text characters.

☐ FRAME-UP™

FAST APPLE DISPLAY UTILITY by TOM WEISHAAR
$29.50: Includes Peeks & Pokes Chart

PROFESSIONAL PRESENTATIONS: Turn existing Hi-Res, Lo-Res & Text pix into Apple "slide shows". *FAST* 2½-sec. hi-res loading! Paddle or Keyboard-advance frames.
UNATTENDED SHOWS optional— each picture arranged and pre-programmed to display 1 to 99 seconds. Text Screen Editor creates black & white text "slides"; lets you key-in type "live" during shows. Mail copies of presentations-on-disk to friends & associates (or home to Mom!).


8 oz.
8 oz.
Apple Mechanic's hi-res type routines and fonts are usable in your programs WITHOUT LICENSING FEE. Just give Beagle Bros credit in your documentation and disk title screen.
CALL-1184

☐ APPLE MECHANIC™

HI-RES SHAPE EDITOR & FONTS by BERT KERSEY
$29.50: Includes Peeks/Pokes Chart & Tip Book #5

SHAPE EDITOR: Keyboard-draw hi-res shapes for animation in your Applesoft programs. Easy "List & Learn" Applesoft demos teach how to do hi-res animation as well as professional-looking hi-res Charts and Graphs.
HI-RES FONTS: Access & create proportionally-spaced hi-res type; each character totally re-definable. Six complete fonts are included on the disk (4 large & 2 small).
MORE: Useful music, text and hi-res tricks for your Applesoft programs. Clear educational documentation and step-by-step instructions for writing graphics programs.

APPLE MECHANIC HI-RES

☐ TYPEFACES™

26 FONTS FOR APPLE MECHANIC by BERT KERSEY
$20.00: Includes Peeks & Pokes Chart

26 NEW FONTS for use with Apple Mechanic programs. Many different sizes and typestyles, both ordinary and *Artistic.* Every character—from A to Z to "★" to "□"—of every typeface—from "Ace" to "Zooloo"—is re-definable to suit your needs. All type is *proportionally spaced* for a more professional appearance. People *do* notice the difference!

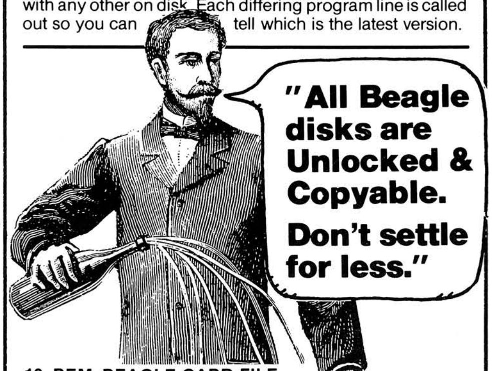
Disk Librarian

NEW!

☐ FATCAT™

PERSONAL DISK LIBRARY by ALAN BIRD
$34.95: Includes Peeks & Pokes Chart

MULTIPLE-DISK CATALOG: FATCAT reads all of your DOS 3.3 and ProDOS™ disks into one or more "Master Catalogs" that can be searched, sorted and printed. Update at any time by simply reading in new or altered disks.
ALPHABETIZE FILE NAMES: Sort your DOS 3.3 and ProDOS disk catalogs alphabetically, by file name, type, etc., to make files easier to find. Re-locate individual file names too. Find files *fast* every time you CATALOG. This feature alone is worth the price of FATCAT!
COMPARE FILES: Compare any program in memory with any other on disk. Each differing program line is called out so you can tell which is the latest version.


"All Beagle disks are Unlocked & Copyable. Don't settle for less."

```
10 REM  BEAGLE CARD FILE
20 HGR: HGR2: POKE 232, 120: POKE 233, 64: POKE
   16504, 7: SCALE=80: P=16: X=99: FOR R=0 TO 31:
   P=P*-1: POKE 230, 48+P
30 FOR Y=0 TO 1: ROT=ABS(64*Y-R): HCOLOR=3:
   FOR A=1 TO 25: DRAW 1 AT X+2*A, X: NEXT
40 HCOLOR=0: DRAW 1 AT X, X: ROT=32: DRAW 1:
   DRAW 1 AT X, X: NEXT Y, R
50 FOR A=0 TO 20: FOR B=0 TO 1: POKE 49237-B,0:
   X=PEEK(49200): FOR C=1 TO 6*A
60 NEXT C, B, A: GOTO 50
```

"APPLE" is a Registered Trade Mark of You-Know-Who.

Beagle Bag!

☐ BEAGLE BAG™

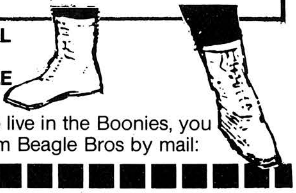
12 APPLE GAMES ON DISK by BERT KERSEY
$29.50: Includes Peeks & Pokes Chart
Apple software customers recently voted BEAGLE BAG to Softalk's "Most Popular Software" list for 1983.

COMPARE BEAGLE BAG with any one-game locked-up disk on the market today. All 12 games are a blast, the price is right, the instructions are crystal clear, *and* the disk is COPYABLE. You can even *change* the programs or list them to learn programming tricks by seeing how they work.
TWELVE GAMES that *last*— TextTrain, Wowzo, Magic Pack, Buzzword, Slippery Digits, and many many more...
EXCELLENT REVIEWS—Read Jan-83 *Softalk*, pg.148.

Beagle Bros
Micro Software Inc.

3990 OLD TOWN AVENUE, SUITE 102C
SAN DIEGO, CA 92110 / 619-296-6400

Attention Dealers: Beagle Bros products rank consistently high on software best-seller charts, and captured FOUR spots on *Softalk's* 1983 Top-30 Most-Popular List (all categories). Telephone any Apple software distributor for fast delivery.



GOTO YOUR LOCAL APPLE SOFTWARE STORE FOR BEAGLE BROS PRODUCTS.

Or, if you happen to live in the Boonies, you can buy directly from Beagle Bros by mail:

RUSH the following disks by First Class Mail:

☐ Alpha Plot	$39.50	☐ Fatcat	$34.95
☐ Apple Mechanic	29.50	☐ Flex Type	29.50
☐ A.M.Typefaces ..	20.00	☐ Frame-Up	29.50
☐ Beagle Bag	29.50	☐ GPLE	49.95
☐ Beagle Basic	34.95	☐ ProntoDOS	29.50
☐ Beagle Graphics	59.95	☐ Silicon Salad	24.95
☐ DiskQuik	29.50	☐ Tip Disk #1	20.00
☐ DOS Boss	24.00	☐ Triple-Dump	39.95
☐ Double-Take	34.95	☐ Utility City	29.50

AT YOUR APPLE DEALER NOW!
Or order directly from Beagle Bros.

Visa/MasterCard or COD, call TOLL-FREE
Orders only / ALL 50 STATES / 24 Hours a Day
1-800-227-3800 ext 1607

OR mail U.S.Check, Money-Order or Visa/MCard# to:
BEAGLE BROS, Dept. R
3990 Old Town Avenue, Suite 102C, San Diego, Ca 92110

NAME ____________________
ADDRESS ____________________
____
STATE & ZIP ____________________
VISA/MC# ____________________
EXPIRATION ____/____ PLEASE SIGN IN MARGIN→

Add $1.50 First Class Shipping, Any-Size Order.
Overseas add $4.00. COD add $3.00. California add 6%.
ALL ORDERS SHIPPED IMMEDIATELY.

Ask the Wizard

The Wizard of Fairhill Castle

Dear Wizard:

I write a lot of programs, and some of them are pretty big, so I need to renumber them from time to time. Every once in a while, something strange happens that I don't understand: some of my numeric constants get changed. Can you tell me why or provide a fix?

Apple has a habit of sometimes releasing programs that are "almost" bug-free. Renumber is one of them. The bug itself is obscure, and comes into play *only* when a value that corresponds to a line number follows a multiply token (*).

For those interested, the reason is that when Renumber is run, it relocates itself in memory, according to the size of the host computer. In the renumbering process, it must search for tokens such as LIST, GOTO, and GOSUB, etc., so that it may renumber these references. It so happens that a couple of these tokens, when disassembled, look like relocatable code to the relocation routine, thus it attempts to "relocate" what are actually Applesoft command tokens.

The fix is easy. You just load Renumber (do *not* run it), make a couple of POKEs, and save it.

```
UNLOCK RENUMBER
LOAD RENUMBER
POKE 4789,172
POKE 4790,171
SAVE RENUMBER
LOCK RENUMBER
```

These are all direct commands that you type from the keyboard. That's all there is to it.

Dear Wizard:

I'm a real game freak, and I like what what you do in your magazine, but it's not enough for me. Where can I get more information?

There are two magazines that I know of that are devoted entirely to games:

Computer Gaming World
1337 North Merona St.
Anaheim, CA 92803

St. Game
P.O. Box 60
North Hollywood, CA 91603

Dear Wizard:

I've seen programs around that do things like showing deleted files when you catalog a disk, so there has to be a way I can do that myself... or can I?

Easy as (Apple) pie. There is an instruction in DOS that tells the catalog handler "skip this file — it's deleted." We can defeat that instruction by "NOPing" it out. [A NOP is an assembly language instruction that means literally "do nothing" except go to the next instruction. It is represented by the hex byte $EA.] You can do a couple of POKEs from the keyboard, or you can put it in the form of a short program or an EXEC file. I'll give it to you in the form of a program for a 48k Apple:

```
10  POKE 44505,234 : POKE
    44506,234
20  PRINT CHR$(4);"CATALOG"
30  POKE 44505,48 : POKE
    44506, 74
```

Notice line 30 is important to include, because it restores your DOS to normal after you have displayed deleted files. Incidentally, when you see a deleted file on the screen, there will be an inverse character about 30 characters after the beginning of the title. This represents the track number of the track sector list for the deleted file. It can be handy if you attempt to recover the file. Sometime we'll tell you how to do that, too.

Nobody asked, but I'll Tell:

What with all the Magic Apples that seem to be arriving here at Fairhill, I thought I'd pass along a quick hint that will let you change your disk drive access without doing a CATALOG or VERIFY or anything. Again, one simple POKE does it, and it would be best to include this in your Hello program.

```
POKE 43624,2
```

Now your next disk access will be to drive two. To change it back, POKE 43624 with a 1. Either POKE can be done at any time from within a program, whenever you want to change the next drive access. Remember, these POKEs are only for a 48k or greater Apple.

Thanks to those who wrote, and remember no question is too tough or too easy for the Wizard of Fairhill Castle. If you have a problem with your Apple, write to the Wizard of Fairhill Castle, P.O. Box 582-AA, Santee, CA 92071, and I'll try and help you solve your problem.

IF/THEN/ELSE

Spells and Potions

©84 L. Anderson

Ralph H. Swerdlow, M.D.

Well, I thought, if my APPLE is so wonderful, why not let it make decisions for me. I always have decisions to make, and a computer is perfect for this task!

Okay, let's ask the computer something. The weather is becoming nice, maybe I should go swimming today. That is a good thought; what does my APPLE say?

So I started up my trusty APPLE and typed in "SHOULD I GO SWIMMING TODAY" and of course a RETURN. Can you guess what happened?

You are right; my APPLE printed "SYNTAX ERROR" because this is not a proper BASIC command! It became obvious that I needed some sort of program to pursue this idea.

Here is my first program:

```
10  TEXT:HOME
20  PRINT "SHOULD I GO SWIM-
    MING TODAY?"
```

Type in this program and then run it. Surprise, for the APPLE prints "SHOULD I GO SWIMMING TODAY?" It doesn't answer a question at all.

Do you see why this program did not answer any question. The program did not ask a question, only instructed the computer to print a statement, which in this case looked like a question. Any characters preceded by the word "PRINT" and surrounded by quotation marks will be printed to the screen or to your printer if it is on. Our computers, while very versatile, have no native intelligence, and do just as they are instructed to do. You and I are smarter, and we are able to recognize the statement in line 20 as a question, but to our APPLES, any string enclosed by quotation marks is a series of ASCII characters to be printed.

Interestingly enough, even if our APPLE was clever enough to recognize this as a question, it could not answer because it doesn't have enough information to make a decision. This is a function of a human, and a realistic answer requires some sort of basis on which to make a judgment!

For example, if I asked you if it was time to eat lunch, you could not answer without knowing the time. Obviously if it was near noon time, the answer would more likely be "YES" than if the time were midnight.

This does not mean that a computer can not make decisions, because they can do so, given some information on which to act. No, they do not do any actual "thinking," but merely compare two values which are placed in the computer's memory by the programmer. Then, depending on these values being equal or not equal, the computer prints an answer which has been supplied by the programmer.

Here is a program to demonstrate this process. It also contains a very little known "CALL" that enables a programmer to erase a complete line from the screen.

```
10  REM IF-THEN BY RALPH H
    SWERDLOW MD
20  TEXT:HOME
30  PRINT "SHOULD I GO
    SWIMMING?"
40  PRINT : PRINT "YOU SHOULD
    IF IT IS WARM ENOUGH"
50  PRINT : PRINT "PLEASE TELL
    ME WHAT IS THE
    TEMPERATURE"
60  PRINT : INPUT "ENTER
    TEMPERATURE —>";DEG
70  VTAB 10
80  IF DEG > 75 THEN PRINT
    "SOUNDS NICE. DO GO
    SWIMMING"
```

Line 10, of course is our standard REM statement, which should always be included in every program. Line 20 sets the screen to full width and then clears it. The main body of the program starts with line 30.

This line first issues a line feed with the "PRINT" command. Note that several commands can be placed on one line if the commands are separated by a colon. The remainder of the line then prints a string of characters which spell out "SHOULD I GO SWIMMING?" The computer does not know this is a question; to our APPLE, this is just a string of ASCII characters. We know it is a question, because we are accustomed to the grammar and form, and by the question mark at the end.

Lines 40 and 50 are the computer's answers. We have, of course programmed the replies, but now it appears as if the APPLE is responding to our question. Line 60 asks us to tell the computer what the temperature is by the INPUT command.

Line 70 cause the answer to be printed on row 10 of our screen, and line 80 then looks at the temperature value the user typed.

Line 80 compares the temperature typed by the user with the value of "75" and if the given temperature is greater than 75, the computer will print out, "SOUNDS NICE. DO GO SWIMMING". This does make it look like the computer is making some logical decisions, but our APPLE is only responding to a program.

Suppose, for a minute, that you prefer to not swim unless the temperature is warmer than 85. Easy enough to reprogram. Just go back to line 80 and change it to "IF DEG > 85. . .", and it is done.

So far, so good. We now have a program that "makes decisions," using the "IF-THEN" command, but Applesoft lacks an "IF-THEN-ELSE" command that is available in other programming languages such as Pascal.

Let us compare these two commands:

IF-THEN

1. Allows the user to evaluate two strings or values. (Yes, strings can be compared.)
2. If the comparison is TRUE, then the remainder of the command is executed, but if the comparison results in FALSE, the remainder of the command line is ignored!
3. Some examples:

```
IF A>40 THEN PRINT
"GOOD"
IF A>=99 THEN PRINT
"SORRY"
IF X<>1000 GOSUB 350
IF S$="JULY" THEN PRINT
"THE MONTH IS ";S$
```

All of these examples can be tried in the immediate mode by merely typing them, followed by a return. Just give a value to A or to S$ and then type in the "IF-THEN" part. Please note that for strings, quotation marks must be used.

IF-THEN-ELSE

1. The computer first looks at the value or string and compares it with that given.
2. If TRUE, then next part of the command is executed just like the "IF-THEN"
3. If FALSE (OR NOT TRUE), the next part of the command will be performed.
4. Some examples:

```
IF A=>100 THEN PRINT
"GREATER THAN 100" ELSE
PRINT "LESS THAN 100"
IF N$="SANTA CLAUS"
THEN PRINT "IT IS DECEM-
BER" ELSE PRINT "GO TO
THE STORE AND BUY
YOUR OWN PRESENTS"
```

Our computers can be programmed to behave as if they had the "IF-THEN-ELSE" command in a very simple fashion. Here is a modification of the last program to show how:

```
10  REM IF-THEN-ELSE BY
    RALPH H SWERDLOW MD
20  TEXT:HOME
30  PRINT "SHOULD I GO
    SWIMMING?"
40  PRINT : PRINT "YOU SHOULD
    IF IT IS WARM ENOUGH"
50  PRINT : PRINT "PLEASE TELL
    ME WHAT IS THE
    TEMPERATURE"
60  PRINT : INPUT "ENTER
    TEMPERATURE - > ";DEG
70  VTAB 10
80  PRINT "NO, DON'T GO
    SWIMMING. IT IS TOO COLD"
85  END
90  IF DEG > 75 THEN VTAB 10:
    CALL - 868: PRINT "SOUNDS
    NICE. DO GO SWIMMING"
```

Lines 10 to 70 are the same, but line 80 is new. This line prints, "NO, DON'T GO SWIMMING. IT IS TOO COLD" on row 10. Note the END in line 85.

Run this program, and observe that "NO. . ." will be printed to the screen. This is our "ELSE"!

If line 85 were to be taken out, the program would continue to Line 90 which compares the temperature inputed by the user in line 60 with the value of "75".

Line 90 sets the output of the string "SOUNDS NICE. . ." to row 10, and will overwrite the statement, "NO, DON'T. . ." printed by line 80. But what if the statement in line 80 is longer than the statement in line 90. Try this as an experiment and see for yourself what happens. The beginning of line 90's output replaces that of line 80, but the last part of line 80's output is still present on the screen, making a very confusing sentence! What we would like to do is "erase" the entire string produced by line 80!

This could be done several ways, for example by printing a string consisting of 40 blanks. This type of command would look like this.

```
VTAB 10:FOR N = 1 TO 40:PRINT
" ";:NEXT
```

With a blank between the quotation marks. However, this adds another line to the program. Note the semi-colon. This concantenates (joins together) all of the blank spaces on one row of the screen and effectively erases that row.

Here is another way to do the same thing. Add this line to your program:

```
75  S$ = "NO, DON'T GO
    SWIMMING. IT IS TOO COLD"
```

Change line 80 to read:

```
80  PRINT S$
```

Change line 90 to:

```
90  IF DEG>75 THEN VTAB 10:
    FOR N = 1 TO LEN (S$): PRINT
    " ";:NEXT N:PRINT "SOUNDS
    NICE. DO GO SWIMMING"
```

This does the same job, but only prints as many blanks as there are characters in S$.

There is an even easier way to do the same thing, and that is by using "CALL -868" as I did in the original line 90. This is a machine language call, that goes to a sub-routine in the monitor of our computer that clears everything from the cursor's position to the end of that row. Using this call, the entire row is "erased" in one graceful swoop, and much faster that printing blanks! We are not limited to choosing between only two alternatives, but can ask our computers to "consider" several possibilities. Here is a program that demonstrates how:

```
10  TEXT:HOME
20  PRINT "SHOULD I GO
    SWIMMING TODAY?"
30  INPUT "ENTER
    TEMPERATURE : ";DEG
40  INPUT "IS IT SUNNY OR
    RAINY OUTDOORS S OR R
    ?? ";W$
50  VTAB 10
60  PRINT "DON'T GO SWIMMING
    TODAY"

70  IF DEG >75 AND W$ = "S"
    THEN VTAB 10:CALL -868:
    PRINT "SOUNDS LIKE A NICE
    DAY TO SWIM"
```

In this program, line 70 requires that BOTH DEG>75 AND S$="S" be true or only "DON'T. . ." will be printed.

These few tricks enable an Applesoft programmer to have the "IF-THEN-ELSE" easily available in their programs. With just a little bit of practice using this technique, you will have no trouble writing more concise programs that require decision making. No longer will you be forced to employ GOTO or GOSUB to control program flow. "IF-THEN-ELSE" is a very useful addition to our Apple's capabilities.

Uncle Bill's Column

Uncle Bill

Off We Go!

Hi kids, I'm Uncle Bill. This column is dedicated to having fun with your Apple computers, making lots of money, becoming famous and seeing what you can get away with while your parents aren't looking. Also, I'll be happy to pass on several tips, hints, gossip, scandal and whatnot. Some of the issues you will all be interested in include software piracy, illegal entry into bank accounts, electronic grade changing and burning down your school with a modem. Naturally, I'll include some tips on how to meet girls and tap telephones.

Getting Weird With the CHR$ Function

Inside your Apple is something called ASCII-Code. (Say As-Key.) They say it stands for American Standard Code for Information Interchange, but I think it is a good way to be sneaky with your computer. For example, let's say you want to pass a note to your friend in class, like "Let's ditch school and go work on our Apples." Now if you get caught writing a note like that, you'll get hollered at, and they may even keep you after school. However, using the CHR$ function, you can get the message across in the guise of passing a computer program. Let's see how it works.

Each character on your keyboard has a corresponding ASCII code. For example, CHR$(65) is the letter "A". By figuring out what CHR$ value is associated with what letter, you can put everything into code that only you and kids with computer smarts can understand. Since the ASCII values are in sequential order of the alphabet, it isn't too difficult to figure out what code goes with what letter. Thus, since CHR$(65) is "A", CHR$(66) would be the letter "B", CHR$(67) the letter "C" and so forth. That means all the upper-case letters are in a range from 65 to 90. Lower-case letters range from 97 to 122.

So far so good, but how do you use CHR$ in programming? That too is as simple as pie. You use the PRINT statement just as you would with text, but you do not need the quote marks. Try the following:

```
PRINT CHR$(65)
[RETURN]
```

You will get the letter "A" printed on your screen. Now, let's put it in a program and see what we get.

```
10  TEXT:HOME
20  PRINT CHR$(83); CHR$(69);
    CHR$(88)
```

Run that little sucker, and you'll see what I mean about the usefulness of CHR$ codes. You will notice how we used semi-colons to separate the different CHR$ values (letters) in the program. If we had used commas instead, the message would have been spread across the screen.

But there's more you can do with CHR$. For example, let's say you want to print the message "NERD" using the quotation marks. Any jerk can PRINT "NERD", but they get NERD instead of "NERD." However, using CHR$(34), you can print quotation marks to your screen. Try the following program:

```
10  TEXT : HOME
20  PRINT “NERD”
30  PRINT CHR$(34) + “NERD” +
    CHR$(34)
```

Notice how we used the plus sign (+) to tie everything together and mix text with the CHR$. That's called concatenation, just in case you didn't know. (When you forget to do your homework, tell your teacher it wouldn't concatenate on your computer.)

(Warning: The County Veterinarian has determined Uncle Bill's Column to be hazardous to your health.)

Publisher's Note: Please do not send Uncle Bill any money for his various schemes. Those of you who sent in $500 for Uncle Bill's Modem User's Permit have all been returned your money. There is no charge for or special permit necessary for communicating via modem. Uncle Bill's parole officer has been notified.

Another secret use of the CHR$ function is to access control characters. When you press the CTRL key along with certain letters, invisible control letters are sent to your screen. Since they're invisible, it is difficult to use them in programming. For example, [Ctrl–G] is a bell, and you can use a descriptive string variable to place it in your program. Try the following little program:

```
10  TEXT : HOME
20  BELL$ = CHR$(7) : REM 7 IS
    THE ASCII VALUE FOR
    CTRL-G
30  PRINT BELL$
```

Use that whenever you want to ring a bell in your programs.

If ringing bells isn't your idea of a good time, you might be interested in accessing your disk drive from a program. (Look kids, if you want to be a real hacker, you gotta learn how to access your disk. Right? So pay attention.) From a program, to make your disk work, you need to use [Ctrl–D] or CHR$(4).

Since your disk access can be anywhere in a program, it is also a good idea to include a carriage return. But how do you get your program to hit the

RETURN key? With CHR$ it is a snap. The ASCII code for a carriage return is 13. Since you want to have the carriage return before your disk command, CHR$(4), define a string variable as D$ = CHR$(13) + CHR$(4). Then when you PRINT D$, you get both your carriage return and your disk command. Try this next little program.

```
10  TEXT : HOME
20  D$ = CHR$(13) + CHR$(4)
30  PRINT D$ "CATALOG"
```

Use that program as your HELLO program on your disk, and whenever you boot your system, it will automatically CATALOG it for you.

Code Writer

Now if you're going to write secret messages, you don't want to spend all your time looking up the ASCII values. To find the ASCII value of a key, use the ASC function. For instance, if you PRINT ASC("A") you will get 65. The ASC function is just the opposite of the CHR$ function. You give ASC a character, and it gives you an ASCII value, and you give CHR$ a number, and it gives you a character. (That's really neat, huh?) What we need is a program that will encode our messages. Whenever you press a key, it will produce the letter and the ASCII value for it. Once we have all the values, we can enter those into DATA statements and pass all those notes mentioned at the beginning of the column.

Code Writer

```
10  TEXT : HOME
20  X = 1 : Y = 1
30  GET A$
40  IF A$ = CHR$(35) THEN END :
    REM CHR$(35) IS THE # SIGN
50  VTAB Y : HTAB X : PRINT A$;
60  VTAB Y + 1 : HTAB X : PRINT
    ASC(A$);
70  X = X + 3
80  IF X >39 THEN X = 1 : Y = Y
    + 2
90  GOTO 30
```

When you RUN Code Writer and key in your message, you will be given the ASCII value below the individual letters. Copy those values down after you have written your message and use Decoder to read the message. The last data value should always be a zero so the program will know when to end.

Decoder

```
10  TEXT : HOME
20  READ X
30  IF X = 0 THEN END
40  PRINT CHR$(X);
50  GOTO 20
100 REM ******************
110 REM PUT YOUR DATA HERE
120 REM ******************
130 DATA 83,69,78,68,32,85,78,67
140 DATA 76,69,32,66,73,76,76,32,77
150 DATA 76,76,32,89,79,85,82,32,77
160 DATA 79,78,69,89,0
```

The decoder program simply READs all of the ASCII values from the DATA statements and PRINTs out the CHR$ held in variable X. As it sequentially goes through the DATA, the value of X keeps changing to reflect the next ASCII value for the CHR$ to be printed to the screen. Run the above program and see the secret message it contains. (Do it while your parents are out of the room, though.)

Well, that's all for this time. Next month, we'll get started on software piracy. Naturally the first thing we'll want to learn is how to keep some geek from stealing our programs. We'll see how to mess up disks, prevent listings and even how to prevent your little sister from running your program.

WORD SEARCH

Q W O R D F
S E A R C H
S T E C H I
B B A K J L
P D M O X Z
F U N Q G N

WORD MATCH

___ ATMOSPHERE	1. WATER
___ BIOLOGY	2. MICROSCOPIC ORGANISM
___ BACTERIA	3. A MATHEMATICIAN
___ NEWTON	4. THE STUDY OF LIVING THINGS
___ OXYGEN	5. 14.7 POUNDS/SQ.INCH
___ LIQUID	6. COLORLESS AND ORDORLESS GAS

WORD SCRAMBLE

1. EJOYN	TERMS
2. BRSEALMC	SCRAMBLE
3. MSTER	ENJOY
4. GAAMANR	WORDS
5. YEAS	EASY
6. CLOHSO	ANAGRAM
7. WDORS	SCHOOL

TEACHER BENEFITS:
- ANY subject ANY Grade Level
- Absolutely no programming knowledge necessary
- Save 20 lessons/tests to disk
- NO GOOFS: Program traps all errors
- Money Back Guarantee

Special Offer

Buy all 3 programs for $60.

Save $25.

Act Now!

Word Search	$34.95	______
Word Match	$24.95	______
Word Scramble	$24.95	______
All 3	$60.00	______
Demo Disc	$ 7.00	______
Shipping	$ 3.00	______
Tax (CA only)	6%	______
Total	$	______

Visa/MC # ______

TEACHER COMMENTS:

"As a group, Word Search, Word Match and Word Scramble is the best teacher software on the market today."
"Without a doubt the most useful teacher utilities at an affordable price."
"Superior, super quality and so easy."

HI TECH of Santa Cruz
126 Lighthouse Ave.
Santa Cruz, CA 95060

Meet the Turtle

Logo Lingo

Jeff Sandys

Hi–res screens, like the one on your Apple, were very expensive and rare ten years ago. The computer terminal was a noisy teletype and drawings were made on plotters. Logo used a robot with a pen to make drawings. These dome shaped robots were called turtles. You can still buy floor turtles to use with Logo.

We don't use the floor to make a drawing anymore. Thanks to the Apple computer, we can draw on the screen. To see your screen turtle, enter CLEARSCREEN. The turtle lives at the center of the screen. To move or turn the turtle use FORWARD, BACK, RIGHT or LEFT, and input the number of steps. The turtle moves and turns at our command:

```
FORWARD 50
RIGHT 90
BACK 100
```

Some other commands that the turtle knows are HOME, PENUP, PENDOWN, SHOWTURTLE and HIDETURTLE. These commands do not use inputs. Give them a try! Many of the commands also have short names. Can you match up the short commands with their long names? These short names will save a lot of typing:

```
FD BK RT LT PU PD ST HT
```

The Turtle's Home

The turtle turns in steps of degrees. There are 360 degrees in a circle or full turn. When the turtle is at home, pointing straight up, its heading is 0 degrees. When it points to the right, the heading is 90 degrees. This is just like a compass with north pointing straight up. The turtle can tell us its heading. The operation HEADING has no input, and outputs the heading. Try this example:

```
HOME RT 123
PRINT HEADING
LT 33
PRINT HEADING
```

From home, the turtle can move forward about 110 steps before it goes off the top of the screen. This is called the Y direction. The SETY command also moves the turtle up and down. To move the turtle left or right, in the X direction, use the SETX command. X and Y are zero at the turtle's home position. We can find out the X and Y position with XCOR and YCOR. Notice how the turtle wraps around the screen in this example:

```
RT 10 FD 2000
HOME CS
FD 100
PRINT YCOR
PRINT XCOR
SETX 50
PRINT YCOR
PRINT XCOR
```

The turtle can be used to explore geometry. The procedure POLY, listed below, draws a polygon with the angle you give it. Can you draw the polygons shown here? Try different angles, less than 180, until you get a polygon. Count the points on the polygon and multiply by the angle used. Is the result 360?

Color by Numbers

Black and white can be boring on a color TV. We can set the background color and pen color to blue, green, orange or violet with the right number. We can either remember which number is which color or use the procedures below to call the colors by name.

Apple Logo has a different syntax than Terrapin and Krell Logo. To enter the program, type in the Main Listing and the listing for your Logo. Try out the different colors:

```
SETBG VIOLET SETPC GREEN
POLY 80
CS SETPC ORANGE POLY 80
CS SETBG
BLUE POLY 80
CS SETBG BLACK POLY 80
SETPC BLACK POLY 80
LIGHTSHOW
SETBG WHITE POLY 80
LIGHTSHOW
```

Did you notice that some colors do not work together? Were there any weird effects? The Apple hi-res screen will not allow certain colors to mix. Blue and orange will not mix with green and violet. The procedure MESSAGE will write a message on the hi-res screen. But the screen ends up black and the message is invisible. Can you figure out a way to make the message visible? Write and tell me how you did it.

Using the turtle to make pictures is a lot of fun. Sometimes it is hard to figure out how to make a drawing. It can help to pretend that you are the turtle. Ask yourself, "What would I do to draw this?' If your drawing program gets too confusing, break it into small procedures. Learn the angles of a triangle, square and star, then write a procedure for each of these polygons. By keeping your procedures small, your program will be easy to debug.

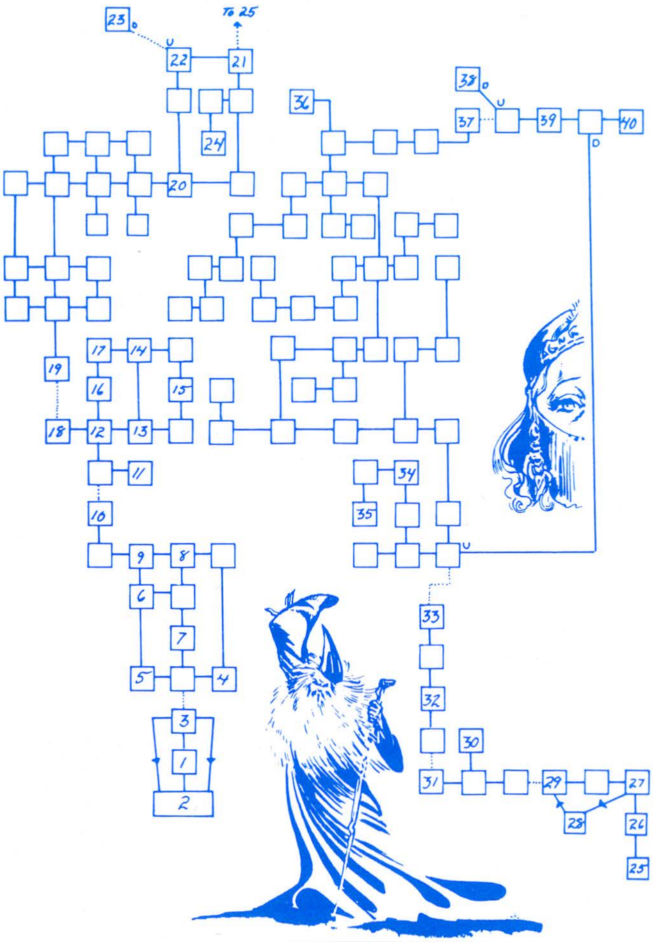
Logo Lingo
P.O. Box 30668
Seattle, WA 98103

MAIN LISTING, for all Logo's

```
TO BLACK
OUTPUT 0
END

TO WHITE
OUTPUT 1
END

TO GREEN
OUTPUT 2
END

TO VIOLET
OUTPUT 3
END

TO ORANGE
OUTPUT 4
END

TO BLUE
OUTPUT 5
END
```

```
TO POLY :ANGLE
PU
HOME CS BK 50 LT
      (180 -:ANGLE)/2
PD HT REPEAT 50 [FD
       100 RT :ANGLE]
PU HOME ST
END

TO LIGHTSHOW
REPEAT 10 [SETBG 2 SETBG
3 SETBG 4 SETBG 5]
SETBG 0
END

TO MESSAGE
SETBG BLACK SETPC ORANGE
HOME CS HT PG2 PU
WRITE :MESSAGE SETPC BLACK
PU HOME
WRITE :MESSAGE
PU HOME ST PG1
END
```

Wizard and the Princess



```
MAKE "MESSAGE [[-120 100] [-120 50]
        [-92 35] [] [-80 85] [-66 35]
        [-52 50] [-52 85] [-66 100]
        [-80 85] [] [-12 85] [-26 100]
        [-40 85] [-40 50] [-26 35]
        [-12 50] [-12 65] [-22 72] []
        [2 85] [16 100] [30 85]
        [30 50] [16 35] [2 50] [2 85]
        [] [-40 10] [-40 -55] [-12 -70]
        [] [0 -5] [0 -55] [] [15 -70]
        [15 10] [45 -70] [45 10] []
        [88 -5] [74 10] [60 -5]
        [60 --55] [74 -70] [88 -55]
        [88 -30] [74 -22] [] [100 -5]
        [100 -55] [114 -70] [128 -55]
        [128 -5] [144 10] [100-5]]
```

APPLE LOGO LISTING

```
TO PG1
.DEPOSIT 49236 0
END

TO PG2
.DEPOSIT 49237 0
END

TO WRITE :LIST
IF
EMPTYP :LIST [STOP]
IF EMPTYP FIRST :LIST [PU MAKE
    "LIST BF :LIST]
SETPOS FIRST :LIST
PD WRITE BF :LIST
END
```

TERRAPIN
and KRELL LISTING

```
TO PG1
SPLITSCREEN
END

TO PG2
TEXTSCREEN
END

TO SETBG :N
BACKGROUND :N
END

TO SETPC :N
PENCOLOR
:N
END

TO WRITE :LIST
```

The Wizard and the Princess:

Shortcut to Adventure

Cassidy/Katz/Lynn/Waisman

Shortcut to Adventure is a series of excerpts from the first and subsequent volumes of "A Shortcut Through Adventureland," by **Jack Cassidy, Pete Katz, Richard Owen Lynn** *and* **Sergio Waisman.** *It was reviewed in the April* Apple's Apprentice, *and is copyright © 1984 by Datamost™, Inc., and is used by permission. We extend our thanks to Dave Gordon, president of Datamost, for his generosity.*

The authors designed *Shortcuts* to help you when you are *really* stuck. You could use it as a complete cheaters guide if you wanted, but then that would take most of the excitement out, wouldn't it . . .

Map Notes

Map on Page 24

North is at the top of the map.

Each box ☐ represents one room or location. An empty box means there is nothing special to do here.

A line with an arrow [→] indicates a one-way passage.

A line with a backward-pointing arrow [←] illustrates that you will wind up in the same place.

A dotted line . . . marks a passage that requires problem solving.

The letter **U** or **D** signifies that the passage goes up or down.

The Wizard and the Princess™

In Sierra On-Line's *Wizard and the Princess*, it is the adventurer's goal to rescue the Princess from the evil Wizard and bring her back to safety. This adventure is fairly complicated, mainly because some of the solutions are abstract. In other words, a guessing game might just lead to success, while a logical approach might lead nowhere.

General Hints

During the first part of the adventure, in the desert, you will randomly encounter rattlesnakes. You may either escape or use the stick from #4 to drive them away.

At any point during the game you may be told that you are thirsty. When this happens, drink water.

Towards the end of the adventure you will arrive at a castle. Once you're there the Wizard will teleport you around when you try to do certain things. Do not get frustrated — just work on killing him (see #38).

Procedures

1. This is the start. There is nothing to do in the town of Serenia. You have to find the Princess and bring her here.

2. Although it is not shown on the map, there is more than one spot here. You should go south and east, looking at rocks, until you find one without a scorpion behind it. (This is randomly determined.) Get the rock (that had no scorpion behind it) and go north and west until you get to #1 or #3. Use the rock at #3

3. You must throw the rock from #2 to kill the snake and continue.

4. A stick can be found here. It is used against most snakes by typing "use stick".

5. Looking at the hole will reveal a cracker. This is used in #14.

6. If you get a rock, the snake will thank you by telling you a magical word. Write it down. This word turns you into a snake for a couple of turns.

7. After using the stick to kill the snake, you may look in the hole to find a note. Look at the note and copy what you see onto a piece of paper (real paper, not in the game).

8. Inside the locket, which is found here, is a magical word which is said at #29.

9 There is another note here. Look at it and write it above the first note (#7). This should form a magical word used at #10.

10. Say "hocus" (see# 7 & #9 for origin). This will allow you to cross the chasm.

11. The apple that is found here will be used at #35.

12. The goblin will steal some of your equipment. You cannot stop this. You must retrieve your things from #14.

The Wizard and the Princess is a registered trademark of Sierra On-Line, Inc.

13. You can go through the hole on the tree, and down the stairs (not shown on map). However, the door cannot be opened from this side.

14. Since the crevice is too small for you to go through, you should say the magic word from #6 (hiss) and turn into a snake. You can then go through the crevice and go south (not shown on map) until you find your equipment. You may go back the same way, or unlock the door and exit through #13.

15. When you give the cracker to the parrot, he will give you a vial which is used at #12.

16. From this brook, you can refill your water supply.

17. You may climb this tree to see the boat which you will be sailing (#19).

18. By giving the bread to the lion, you get rid of him.

19. The rope found here is to be used at #22. Entering the boat, you will notice that it has a hole in it. Use the blanket to cover the hole and you will be able to sail off.

20. After getting the shovel from #23 (see below), dig. You will find a chest, but a pirate will steal it and take it to #24.

21. There is an anchor here which is to be used at #22. Also, after finishing this island, drink the liquid from the vial from #15, and go north to #25.

22. Tie the rope from #19 to the anchor from #21. Then throw the anchor which hooks up to the tree, allowing you to proceed to #23.

23. Get this shovel and dig at #20.

24. After digging at #20, come here and get the chest. Inside you will find a harp to be used at #31.

25. The only way to get here is through the action at #21.

26. This ring is to be used at #38.

27. Talk to the woman to get some information. The first time you go west, you'll go to #28. After the first time, the other west path is taken.

28. If you go to the rainbow, you'll find a coin to be used at #32.

29. Say the word found inside the locket from #8 (Lucy) to go west.

30. You can get all of your equipment by going into this cave.

31. You can get past the giant by playing the harp from #24.

32. Buy a trumpet with the gold coin found at #28.

33. Play the trumpet from #32 to lower the bridge. Your next goal is #38.

34. There is a mistake in the game here. It says there is a doorway to the east and one to the south. The doorways are actually to the west and south.

35. Kill the boar. If the Wizard zaps you here, you will see a boar. To kill him, give him the apple from #11.

36. Escape from the cell. If the Wizard zaps you here, you will be locked in a cell. To escape, say the magic word HISS. This lets you exit freely.

37. To open the door to the east, you must use the knife to pick the lock.

38. Kill the Wizard. If this room is empty when you enter, go out and come back in again. You will see a bird, actually the Wizard in disguise. To kill him, put on the ring from #26 and rub it.

39. This frog is actually the Princess. Kiss the frog to change her back. If the Wizard is still alive, he will zap you to a new location out of this room.

40. Look in the closet for a pair of shoes. Take the shoes and look at them. You will see a magic word to transport you and the Princess back to Serenia. Put on the shoes and say the magic word. When you are back in the town, type another command and the game will end.

The Duncan Letters (from page 13)

about my having the Macintosh. The princess here just returned from a camp in the Silicon Valley where they teach young squires and squiresses how to use these Magic Apples, as well as others. Needless to say, the castle is all astir with excitement and anticipation! Oh, I wish that minstrel would return soon with more apples, so that things would get done around here!

Continue your apprenticeship, Duncan, and learn well. Keep me posted on your progress, will you? With knowledge, you can do anything, remember? Ah, but don't let it get to the point of exclusion . . . remember that book I gave you for your last birthday:

". . . Once I had a brain and a heart also; having tried them both, I should much rather have a heart."

Whatever you do, care enough to do your best . . .

Your friend and cousin,

Joseph

Spanish Verb Conjugator

Tim Graham

Here is a program, submitted by 14-year old Tim Graham of Manhattan Beach, CA, that might help you with your homework. It will break down *AR*, *ER* and *IR* verbs. What it can't handle is irregular verbs. It would seem that perhaps some of Tim's ideas could be applied to other languages. Worth a try, huh?

Incidentally, Tim gets a free 12-issue subscription for his efforts . . . How about you?

```
]LIST

10  REM

SPANISH VERB CONJUGATER
     BY TIM GRAHAM

100  ONERR  GOTO 600
110  HOME :BS$ =  CHR$ (92)
120  INPUT "VERB TO CONJUGATE? ";
     A$
130 B$ =  RIGHT$ (A$,2)
140  ON B$ = "AR" GOTO 200: ON B$
      = "ER" GOTO 210: ON B$ = "I
     R" GOTO 220: GOTO 120
200 YO$ = "O":TU$ = "AS":EL$ = "A
     ":NO$ = "AMOS":ES$ = "AN": GOTO
     300
210 YO$ = "O":TU$ = "ES":EL$ = "E
     ":NO$ = "EMOS":ES$ = "EN": GOTO
     300
220 YO$ = "O":TU$ = "ES":EL$ = "E
     ":NO$ = "IMOS":ES$ = "EN"
300  HOME : PRINT : PRINT
310 C$ =  LEFT$ (A$, LEN (A$) - 2
     )
320  PRINT "        YO "C$;YO$
330  PRINT
340  PRINT "         ";BS$
350  PRINT "        TU "C$;TU$
360  PRINT
370  PRINT "   ";BS$
380  PRINT "   EL  ";BS$
390  PRINT "   ELLA >"C$;ES$
400  PRINT "   UDS./"
410  PRINT
420  PRINT "NOSTROS "C$;NO$
430  PRINT
440  PRINT "  ELLOS";BS$
450  PRINT "  ELLAS >"C$;ES$
460  PRINT "  UDS. /"
470  PRINT : PRINT
500  PRINT "HIT ";: INVERSE : PRINT
     "RETURN";: NORMAL : PRINT "
     TO CONTINUE"
510  POKE 49168,0
520 KY =  PEEK (49152)
530  IF KY = 141 THEN  POKE 49168
     ,0: RUN
540  GOTO 520
600  IF  PEEK (222) = 255 THEN  VTAB
     23: PRINT "...BYE": END
610  VTAB 23: PRINT "ERROR "
620  FOR PA = 1 TO 1000: NEXT PA
630  RUN
```

Direct to you . . .

Hours of fun and learning enjoyment, learning about Apples — the Apple's Apprentice way.

Save $6.00 from the newstand price and subscribe ***now*** to 12 information-packed issues for only $24.00. (24 issues just $46.00 and you're $2.00 better off.)

Apple's Apprentice
P.O. Box 582-AA
Santee, CA 92071
(619) 562-7785

☐ Visa ☐ Mastercard ☐ check
Card No. ______
Exp. date ______

Name ______
Address ______
City ______ State ______ Zip ______

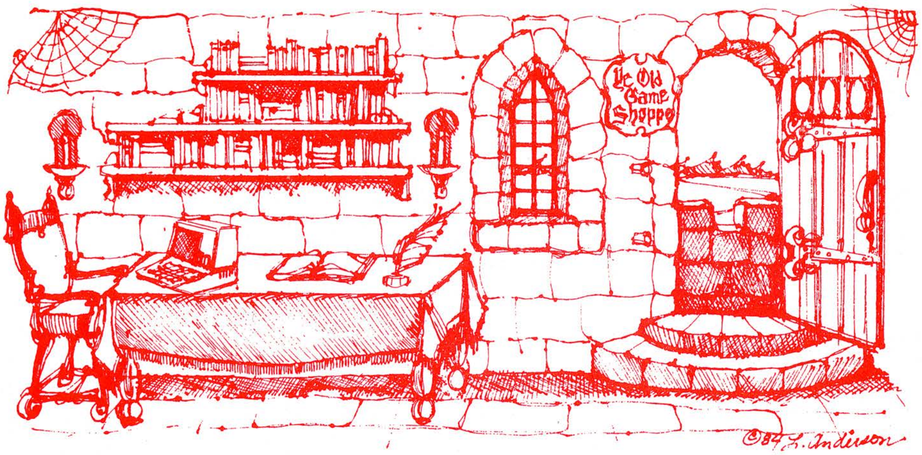
Signature ______
(if minor, parents signature)

☐ Enclosed is $24 for 12 issues
☐ Enclosed is $46 for 24 issues

Game News and Reviews

Ye Olde Game Shoppe

Val J. Golding, prop.



Ah, my young friends, I was hoping you would pass this way again soon; I have many new games to offer you. Master Michael, I can not help but notice your brother Matthew is not with you this day. However, I know this is the morn of the Tuldurry Knights Tournament, and I suspect he is eager to attend. I myself plan to be there for the grand championship later today, so we must not allow pleasantries to intercede with our offerings. If you have no other plans, perhaps you may wish to ride with me to the tournament.

Bezare is Bizarre

One of the most unusual games to come my way ever, *Bezare* purports to be from a planet in the Bezardian system. The publishers (Roger Wagner Publishing) inform me they were so impressed by the game that they spent countless hours converting it to Apple format from a Bezardian computer diskette. After all, bits are bits and bytes are bytes, no matter the language. Converting game instructions proved a more difficult task, so the original Bezardian directions have been reproduced in their native tongue, a text that appears to the eye as a combination of Egyptian hieroglyphics and oriental characters. It therefore falls upon the player to do his/her own translation.

Standing on its own, as an arcade game, Bezare deserves at least a B+, but adding the challenge of discovering on your own *how* to play the game must add at least another full point. I've discovered quite a bit on my own in a limited period. There are three distinct and repeating arcade games, or levels of play.

At this point I have not become fluent in Bezardian, so the current and high scores — which I rely on heavily — are of little help, since they are displayed in Bezardian pictograms. Like any other game, though, practice and familiarity produce the technique required to win.

I've at least figured out that some of the standard arcade commands like "P" for pause and [Ctrl-S] for sound, do work. In addition, should you be playing at the office, a [Ctrl-W] instantly converts your display to a dummy spreadsheet! The graphics are simple but effective. The intrigue of what the game does is as exciting and as effective as the game itself. I don't think this is for the casual game-player, but for anyone above that first elementary level, the challenge is first cabin.

As we go to press, I've just made a deal with the publisher to offer Bezare at a reduced price to Apprentice subscribers only, along with a full-color 17″ x 24″ poster! I'll put a notice elsewhere in this issue.

Bezare
Roger Wagner Publishing
10761-E Woodside Ave.
Santee, CA 92071
$19.95

Hanky-Panky with Hinky Pinky

Now here's a game with a . . . never mind those cobwebs, they're not hurting anything — pay attention now . . . a game sold on the honor system. You pick up a copy of the game from the supplier or a friend; if you like it, you send them 25 dollars, and they send you back a registered copy, as well as future updates. You are encouraged to make copies and give them to your friends, along with the request, of course, that they too send in their money or pass the copy on.

Now, let's see, I haven't told you about the game itself, yet, have I? The Hinky Pinky Game is a rhyming game with a large dictionary containing mated syllable groups. You take a hint and try to guess the hinky pinky rhyme pair. For example, a hint reading "Wizard's abode" might lead to "power tower" or the hint "noisy insect inhabiting a bushy area" could mean "cricket thicket" and so on. This is lots of fun, but the really good part is when you flip the disk over to create your own rhyming pairs. Now you have a complete rhyming dictionary at your command. You can type in a word, then select from the menu a list of words that rhyme with yours. It's great for the budding poet.

The Hinky Pinky Game
The 22nd Avenue Workshop
P.O. Box 3425
Eugene, OR 97403
$25.00 (see story)

Once Over Lightly

More for our puzzled (puzzling?) readers: a trio of word games from *Hi-Tech of Santa Cruz*, $25 each or all three for $60. Word Search, Word Match and Word Scramble. In the first, you locate words in a block of characters. The second matches words to definitions, while the third, does as its name implies. Looks neat.

Chez 21, by *Microlon, Inc.* P.O. Box 1529, San Marcos, TX. A moderately good blackjack game with an excellent tip book (when to split, when to stay, etc.)

Though the day is still young, it is time to shutter the windows; I must be off to the tournament. You, young Michael, may ride with me on my steed If you desire, and all of you are welcome to return at another time. The sorceror tells me he has heard from one of his Hexels that a stranger rides toward our land, his satchels filled with new games and Magic Apples, so do hurry back.

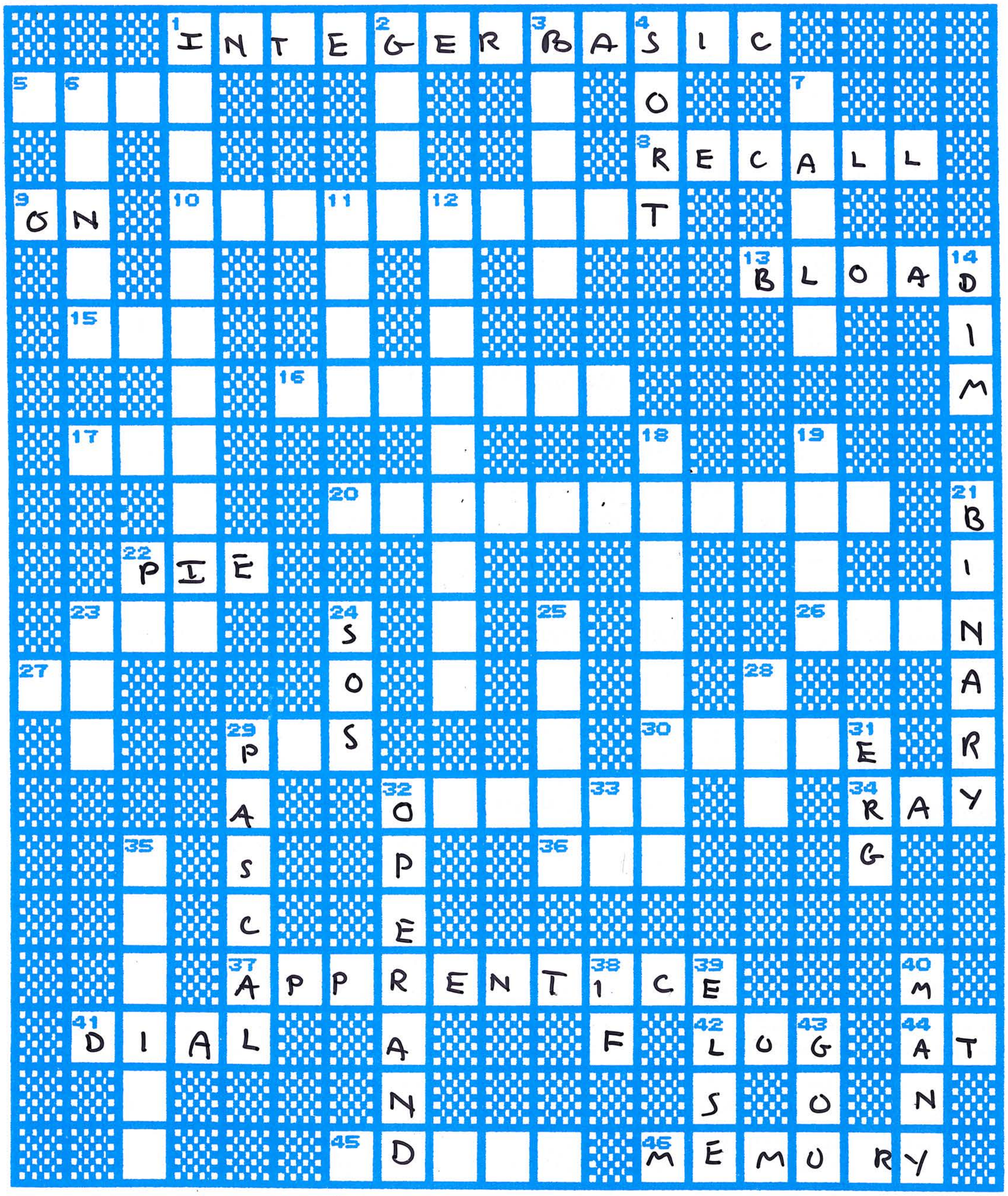
Cross Words

Don't be angry, let's not have words . . .
Unless, of course, they are cross words . . .

Hints

You should read the *Apple's Apprentice* carefully. Many of the answers appear in the stories. Watch out for puns and word plays. For example, the clue "job" might mean "tasc" (the compiler). When you think you have an answer but are not sure, look it up in a dictionary or a BASIC manual.

Answers appear on page 6. Good luck!



Across

1. Came before Applesoft
5. Code word is "sesame"
8. Recollect
9. Condition when bit = 1
10. Held at Tuldurry
13. Command to get a binary file
15. A reader
16. Programmer's program
17. Spoil
20. Take apart
22. Mom's mate
23. What we confess
26. Not now
27. Slang for bad
29. Little used BASIC command
30. Opposite of 13 across
32. Instructs the microprocessor
34. A beam
36. What a lion does
37. A student magician
41. What you do with a telephone
42. Made from a tree
44. Name for the character "@"
45. In graphics, to erase
46. Apples often have 64k

Down

1. What you get in school
2. Goof-off or thug; Alice was one
3. In radio or tv, delete
4. Alphabetize
7. Cartoon character
11. Grows underground
12. Condition of a spoiled apple
14. Not too bright
18. There's a bug in it
19. They are often hatched
21. System with only two values
22. Made from apples
23. This command rhymes with 23 across
24. Distress signal
25. Brand name of bacon
28. Popular soft drink
29. A structured language
31. A measure of energy
32. What most opcodes need
33. In some systems, a loop
35. Known to plunder
38. A test
39. Alternative to THEN
40. Opposite of few
43. A sticky mess

ANNOUNCING

Spellcaster™

The Easiest Computer Language in the World.

Spellcaster is a new computer language that is easier to learn than LOGO or BASIC. With Spellcaster you can learn to program your own fast-action video games, intricate art designs, interactive courseware, and many other applications using graphics, color, movement or sound.

What makes Spellcaster so easy to learn?

- Spellcaster is the simplest language on the market. Young children use it to draw all over the screen. As they grow, the tutorial entices them into true programming.
- Everything a Spellcaster program does leaves marks on the screen. You watch all its inner workings in motion.
- Spellcaster's on-screen tutorial makes your computer teach you programming. It even teaches you how to program your own video games.
- Debugging a Spellcaster program is easy, because you can stop it, make it back up to the mistake (while you watch), change it, and let it run forward again.
- Spellcaster's manual, The Book of Spells, is light-spirited, color-coded and loaded with examples.

For $39.95 you get

The Book of Spells (Manual)

Copyable disk with:
  Spellcaster Language
  Tutorial program
  Video game subprogram library

One issue of The Spellswappers' Gazette *

MC/VISA call 1-703-433-8788

At your dealer or direct from:

Shenandoah Software
1111 Mt. Clinton Pike
Harrisonburg, VA 22801

For Apple II, II+, IIe, IIc,
and Commodore 64

The language: Spellcaster was designed from scratch to be easy to learn, yet formally complete. Its primitives are not numeric operations, but screen operations. Each change in state is visible to the programmer. Spellcaster is highly structured (nested conditions, loops with exit conditions, recursion) but the control structures are expressed with radical simplicity. "Teleporters" are unique language features that partially save and restore process state to permit real-time video game programming.

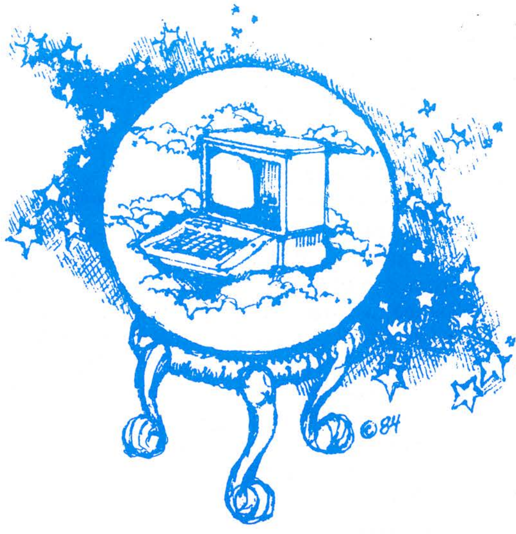
The environment: Imagine an editor and an interpreter so wed that every keystroke, as it is typed, is syntactically checked and executed, so you instantly see its effects. If you backspace, the program reconstructs its previous state — even in the middle of conditions and loops.

The tutorial: Keystrokes generated by the tutorial guide the user, stroke by stroke, through experiments in programming. The tutorial can generate macro's which execute on the spot. The pedogogical approach is to have beginners build their own video games.

* On the flip side of the Spellcaster disk is a free issue of The Spellswappers' Gazette, a diskette magazine of readers' games, comments and programming know-how.

Apple is a trademark of Apple Computer, Inc.

New Products:
The Crystal Ball



Earn-it/Play-it Unique Concept

Earn-it/Play-it from Dynacomp, is an automated "contingency management" program for the classroom. Access to computer games is contingent upon the student's behavior.

Each use of Earn-it/Play-it with a student is called a session, generally one school day. The teacher may identify up to nine objectives for each student during each session (e.g., completes homework, passes quiz, zero misbehavior rate). At the end of a session, the teacher records whether or not the student has achieved each objective. If a specified proportion (say 65%) of the objectives is achieved, the student is rewarded with game credits.

Game credits give the student an opportunity to play one of the games contained on the Earn-it/Play-it disk. The student may choose to save the game credit or use it immediately (if the teacher permits). The program keeps track of each student's game credit balance. To maintain the student's motivational level, new games become available at certain milestones. There are a total of six different games on the disk. At game time, the student can select any available game for which he/she qualifies. Naturally, the more interesting games cost more game credits. Price $29.95 from Dynacomp.

Dynacomp also offers *Coach's Corner* at $29.95, a realistic computer football game which allows you to coach rather than play. Coach's Corner teaches football *strategy* by "first hand" experience, and helps you better understand what is happening on the real field.

Dyslectic Learning Package

Dynacomp's *Primer 83* is a very unique and effective learning package designed for people with impaired reading skills. It was originally written to aid in the rehabilitation of a 30-year old man suffering from "word blindness" or *dyslexia*. He had a brain tumor which, after removal, resulted in selective memory loss. He worked with many traditional excercises with only limited success. With the incentive of sponsorship from John Hopkins University, Primer 83 was developed to handle the needs of this individual and other people having this handicap. Primer 83 worked. It also appears to be effective with individuals having reading problems in general. Primer 83 is the culmination of the combined effort of several people, including the original "student," an expert programmer, a grammaticist, a musician and several others. It is well thought out and tested.

Requirements

- The user *does not* need to have prior experience with a computer or have typing skills.
- A cassette recorder is needed: The instructions given in the manual are also recorded on tape.
- A black and white monitor.
- A paddle connected to port #0 to control display speed
- A printer (optional) to print out test statistics and analyses.
- An ECHO II speech synthesizer (optional) for letter and word prompting. Nice, but not vital

Note that a black and white *monitor* is required. Color televisions do not have the resolution of a B&W monitor. There is no point in *further* handicapping the student with a poor display.

Module Descriptions

Primer 83 contains several program modules. The main ones are:

- **Talkwriter** — a program designed to familiarize the user with the keyboard. If an Echo II is present, the letter will be spoken.
- **Letter Flash, Word Flash, Sentence Flash** — Letters, words and sentences are flashed on the screen. Words come from a 600 word vocabulary of common words. 6000 grammatically correct sentences may be displayed or spoken. Each of these modules have many additional features to aid use.
- **Music, Music Editor** — You may create your own tunes and words: load, save or play a song; add, change, insert or delete a note. Notes range over two octaves and can be full to $\frac{1}{32}$. Four tunes come with the Music module.
- **Sci-Fi Story** — A nearly infinite variety of short science-fiction stories can be created and displayed.
- **Trouble Shoot** — All of the question and answer modules have the option to save statistical results. These include number of presentations, %errors, average response times and specific errors. Trouble Shoot will also analyze specific errors to determine possible perceptual or motor difficulties of rotation, inversion, correct key/wrong hand, etc.

Other Features

- Upper or lower case letters
- Alternate character sets. ASCII, Roman, Collasal and Flow are currently available; you may also design your own font.
- Re-defineable keyboard. Keys may be put in ABC order instead of QWERTY.

Additional information from Dynacomp, Inc. 1427 Monroe Avenue, Rochester, NY 14618, (716) 442-8960, (716) 442-8731.

Contests Galore!

Daze of The Knights

David Sparks

Right now, in county Tuldurry, the Knights Tournament is taking place. Many prizes will be awarded to the bravest and most ingenious of the knights. Here are two more special contests, designed to bring you hours of fun and challenge, while at the same time showing you some of the powerful things your Apple can do as you learn about its internal workings.

We want to hear from you: we want to know how you like the contests; we also welcome your contest submissions. Most of all, we want you to enjoy the *Apple's Apprentice* and have fun learning about computing. Preliminary rules follow; the complete rules and prizes will be published next issue.

General Rules

- Be sure to include your name, address and *age* with each entry.
- Duplicate prizes will be awarded to each of two age groups: under 15 and 15 and older.
- In the event of a tie, the earliest post-marked entry will receive the prize.
- Winner's names will be published in the *Apple's Apprentice* and added to our "honor roll"
- No more than one entry to one contest may be enclosed in the same envelope.
- Envelopes must be marked on the outside with the number and/or name of the contest, and addressed to:

Apple's Apprentice
P.O. Box 582–AA
Santee, CA 92071

Contests 3 and 4

Many thanks to David Sparks, the author of this month's contests.

Type each in, exactly as listed with no space after "REM," save and run, then list. Tell us why they did what they did. Note each program should be saved to disk before running.

Extra Special . . .

If you can write a good explanation that can be understood by others and help them learn about the Apple, we will consider it for publication! And now, it's time to get to work. Good luck!

```
]LIST

10  REM DAZE OF THE KNIGHTS: #3
20 X = 2: POKE 2049,59: POKE 2082
     ,178: RUN
30  PRINT : PRINT "X=";X
```

```
]LIST

10  REM DAZE    OF THE KNIGHTS #4

20 X = 2
30  FOR Y = 2053 TO 2060: READ Z:
      POKE Y,Z: NEXT
40  GOSUB 10
50  FOR Y = 2053 TO 2060: READ Z:
      POKE Y,Z: NEXT
60  PRINT : PRINT "X=";X
70  DATA 88,208,88,202,50,58,177,
     0,178,83,79,32,87,72,65,84
```

TM

The Original Computer Camp

Join us for our 5th exciting season of teaching 7-15 year-olds microcomputer technology in traditional camp settings. We offer the most progressive computer curriculum available, along with complete recreational programs at a variety of ACA accredited camps. Locations include New Hampshire, Connecticut, Pennsylvania, California and Colorado. Courses in LOGO, Pascal, Robotics, Artificial Intelligence, Graphics and more. Write or call our toll-free number today!

1-800-235-6965
(inside Calif. 1-800-824-3349)

The Original Computer Camp, Inc.
559 San Ysidro Road, Dept. AA
Santa Barbara, CA 93108
(805) 969-7871

ACA ACCREDITED CAMP

Crystal Ball (from Page 31)

Home Cataloger

The Home Cataloger is an efficient and easy to use filing and cataloging program, allowing up to 1,500 individual entries (depending on hardware). The user can create customized filing systems, or select one of the program's ten pre-designed cataloging lists. These include telephone list, inventory, travel planner, insurance policies, studies and book lists.

The Home Cataloger has the capability of totaling numbers in any or all numeric categories. Additionally, the program can generate whole lists of selected categories in any order, as well as custom printouts.

Suggested retail is $49.95. For information, contact Continental Software, 11223 South Hindry Avenue, Los Angeles, CA 90045, (213) 417-8031.

Dija Know

Deja vu describes a scene you have seen before — perhaps in your dreams; *Dija know* is our way of describing a simple hint you may have heard before, but forgotten.

Dija know— typed from the keyboard, **FP** will completely re-initialize Applesoft (and wipe out your program).

Dija know— typed from the keyboard, **MAXFILES 3** will reset HIMEM and *will not* wipe out your program.

Dija know— **DEL 1,0**, when typed from the keyboard, will reset LOMEM.

Look for *Dija know*'s in each issue of *The Apple's Apprentice.*

Penguin Scores Again!

	Releases	Hits	Errors
Fantasy	2	2	0
Arcade	1	1	0

Expedition Amazon—A fantasy role-playing game with a sense of humor. Guide your own expedition from Nihil, Texas to Pedro's Trading Post and through the jungles of Peru in search of priceless treasures and the fabled lost city of Ka!

Arcade Boot Camp—Tired of getting 30 seconds of arcade play for your quarter? Face forward, Civilian, and march over to your dealer for this one. Train in five areas vital to arcade skills: Driving, Chopper Flying, Shooting, Jumping & Ducking, and Obstacle Course.

Xyphus—Explore the Lost Continent of Arroya as you develop a band of warriors and spellcasters in preparation for the final confrontation with Xyphus, Lord of Demons! This fantasy role-playing game features four-player independent movement and six separate scenarios, each set in a different region with different types of creatures, weapons, and spells. A true breakthrough in its genre, **Xyphus** is destined to become a classic.

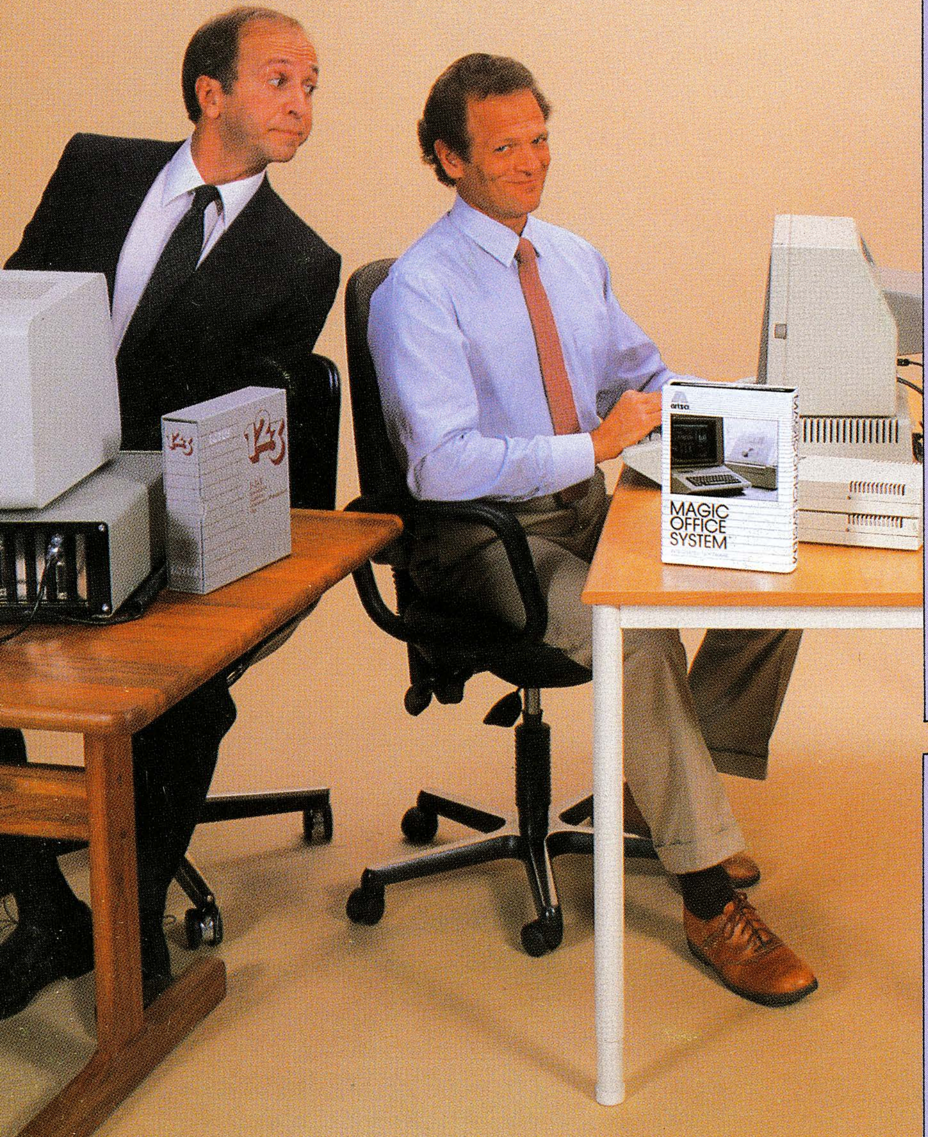
penguin software™
the graphics people

830 Fourth Ave.
P.O. Box 311
Geneva, IL 60134
(312) 232-1984

Expedition Amazon, Arcade Boot Camp, Xyphus, and Penguin Software are trademarks of Penguin Software, Inc.

Apple owners shouldn't be jealous of integrated software like Lotus 1-2-3.

Now you can have something better. Magic Office System.™



Now you can have four of the most powerful programs for the Apple IIe and Apple IIc computers—all in one fully integrated package.

You'll be able to quickly and easily produce letters and reports—complete with spreadsheets and graphics interspersed—with every word spelled correctly.

Using one disk. No shuffle.

WORD PROCESSING: you'll have a greatly enhanced version of an already popular program for creating and editing any kind of document. Printing and formatting are automatic.

SPELLING CHECKER: you'll have the first ever, on-line automatic checker with plenty of room to add words special to your business.

SPREADSHEET: this one is more powerful than VisiCalc and includes variable width columns and output formatting.

BUSINESS GRAPHICS: you'll be able to create pie charts or multiple bar charts to enhance your reports and presentations.

What's especially convenient is how Magic Office System uses icons to simplify your work.

It's all organized into drawers containing file folders, stationery and documents. You can even place folders in other folders, or "cut" parts of documents and "paste" them to others. Or move or copy documents anywhere in the system.

Lotus should be so easy!

Want a Magic Office System immediately? Check with your local dealer. If he doesn't have it, send in the coupon below.

You'll need 64k, an 80-column display, two disk drives and a whole new level of expectation about great software for Apple!

artsci™

I can't find Magic Office System at my local dealer.

☐ Please send me more information right now.

☐ I can't waste time. I want _______ copies of Magic Office System now. Enclosed is $295 per package, plus $19.18 sales tax per package for California residents.

☐ Enclosed is a check for __________.

☐ Charge to my ☐ VISA ☐ MasterCard.

Acct. No. ______________________

Exp. Date ______________

Name ______________________

Business ______________________

Address ______________________

City/State/Zip ______________________

Telephone ______________________

Mail to: ARTSCI 5547 Satsuma Avenue
North Hollywood, CA 91601
Or call: 818/985-2922

AA8

Sorry, IBMers, Magic Office System is only for the Apple IIe and the IIc.

Magic Office System is a trademark of ARTSCI, INC.